Алексей Черных

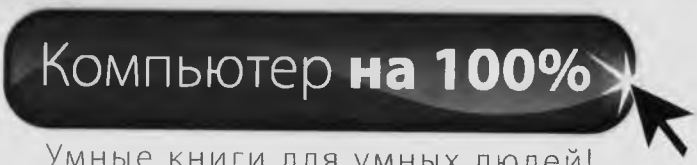
Компьютер **на 100%**

Умные книги для умных людей!

Drupal 7

Бесплатная система управления сайтом

ЭКСМО



Умные книги для умных людей!

Алексей Черных

Drupal 7

Бесплатная система управления сайтом

Москва эксмо 2011

УДК 004.42
ББК 32.973.26
Ч-49

Черных А. Ю.
Ч-49 Drupal 7 / Алексей Черных. — М. : Эксмо, 2011. — 208 с. — (Компьютер на 100%).

ISBN 978-5-699-47059-4

Drupal – это одна из самых популярных систем управления сайтом (CMS). Будучи бесплатной и написанной на популярном языке PHP, Drupal получила самое широкое распространение и признание. Архитектура Drupal позволяет применять данную систему для построения любых сайтов – от интернет-магазинов до блогов, а также произвольным образом оформлять их.

Книга является простым и в то же время достаточно полным руководством по системе Drupal. Для изучения книги и последующего использования Drupal не требуется знания языков веб-программирования. Большое внимание уделено отличиям новой версии системы от предыдущей.

УДК 004.42
ББК 32.973.26

Все названия программных продуктов являются зарегистрированными торговыми марками соответствующих фирм.

Никакая часть настоящего издания ни в каких целях не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами, будь то электронные или механические, включая фотокопирование и запись на магнитный носитель, если на это нет письменного разрешения издательства «Эксмо».

ISBN 978-5-699-47059-4

© Черных А. Ю., 2011
© Оформление. ООО «Издательство «Эксмо», 2011

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

- Описание задачи
- Выбор CMS
- Краткое сравнение
- Выводы
- Цель книги

Случилось так, что мы не смогли больше терпеть свой старый сайт, который создавался в 2000 году, и решили переделать его, воспользовавшись современными технологиями, а заодно подновить, подкрасить (ну, вы понимаете).

Когда мы обратились к специалистам, те спросили: «А какую систему управления сайтом вы используете?» Ответ был простой — никакую. В 2000 году еще не существовало подобных систем, и наш сайт представлял собой старый добротный HTML.

«Тогда возьмите какую-нибудь систему управления, — порекомендовали нам. — Например, Joomla. Все просто, понятно и бесплатно».

Ну, Joomla, так Joomla, решил я и нашел книжку для начинающих. Одолев ее за день, я примерно представлял себе, что такое Joomla. Поскольку сайт у нас был не один, я начал подбирать список модулей для подключения нескольких сайтов, думать, как организовать свою информацию по категориям и разделам, и вдруг наткнулся на занимательную статью:

«Как я делал портал на Joomla и Drupal, или что нужно знать для создания сайта».

Источник: <http://dimoning.ru/o-standartnih-drizhkah.html>

Выводы в статье оказались довольно интересными:

- Drupal: сложно, непонятно, без документации и нормальных шабло-нов. Для новичка вообще кромешный ад;
- Joomla: жить будет, попытаться что-то сделать на ней действительно можно, но не факт, что получится.

Казалось бы, тут и сделать вывод: все, Joomla — это наш выбор. Но что-то меня в этой статье зацепило, и я начал более детально изучать суть вопроса. После пары-тройки часов серфинга я готов был опреде-лить следующее:

- есть множество серьезных коммерческих сайтов на Drupal, и ссылки на них приведены, чего не скажешь про Joomla;
- Drupal декларируется как CMF (Content Management Framework), а не CMS (Content Management System), что круче;
- быстродействие последней версии Drupal выше или сравнимо с по-следней версией Joomla;
- Joomla предоставляет модуль для выполнения функции, а Drupal — набор элементов, из которых можно его собрать;
- Drupal — это религия, а не платформа разработки.

Такие результаты заинтересовали меня еще больше, и я попытался получить более глубокие знания о системе Drupal. В результате доста-точно быстро я понял, что:

- базовым элементом контента Drupal является материал, и весь он пронумерован; меня, как программиста, очень порадовало наличие идентификатора (первичного ключа), в результате чего отсутствует необходимость искусственно разбивать все материалы по разделам и категориям;
- можно описывать различные типы материалов и специфические поля для них — тоже замечательно: можно сказать, что создается практически готовая база данных произвольной структуры;
- таксономия позволяет создать любое количество произвольных классификаторов, то есть сначала формируются материалы, а уже потом они классифицируются различным образом.

Вывод, который я сделал на основании этого, был прост и очевиден:

- Joomla — это система для блондинок, которым надо быстренько сочинить сайтик из страничек;
- Drupal — система для настоящих программистов (если кто помнит, такие люди могли ввести операционную систему в кодах с пульта ЭВМ).

Документация, которая на тот момент была у меня под рукой, состояла, в основном, из технических описаний на английском языке с официального сайта. Она, конечно, охватывала все аспекты и тонкости разработки, но начинающему разработчику было очень тяжело пользоваться ею. Русская документация оказалась не такой полной, и при этом представляла собой перевод тех же технических описаний.

Когда я попытался найти учебник в электронном или бумажном виде, все мои поиски не увенчались успехом. Электронный учебник был не полон, а комментарии к бумажным учебникам оказались, мягко говоря, ругательными, что сразу лишало желания покупать такие книги.

Что было делать? Как говорят, хочешь сделать что-нибудь хорошо — сделай это сам. В результате появился этот учебник, который рождался по ходу осваивания системы и, надеюсь, содержит необходимый набор шагов для новичка, который хочет освоить систему Drupal.

ИДЕОЛОГИЯ САЙТА

- Особенности Drupal
- Составление плана
- Список получившихся шагов

Когда мы прикидывали, как будем переносить сайт на Joomla, все было просто. Достаточно составить список разделов и категорий для текущих страниц сайта, потом создать статьи, которые соответствуют текущим страницам — и все, сайт готов.

Но после изучения основ Drupal все сразу стало гораздо интереснее.

Во-первых, описывать сайт нужно не с точки зрения страниц и их содержимого, а с точки зрения информации, которую мы хотим туда поместить. При этом можно создавать любые типы данных, которые нам нужны. Например, новости, отзывы и т. д. Очень важно начинать именно с данных. Лучше вообще убрать текущую структуру навигации, чтобы она не мешала, и составить плоский список блоков информации. Отдельный блок информации в Drupal называется *материалом*.

Во-вторых, мы можем поддерживать не один сайт, а множество и распределять материалы между ними. Это могут быть сайты как на одном, так и на многих языках, и мы можем распределять материалы по сайтам по нашему желанию. То есть структура хранения материалов полностью отделена от их публикации.

В-третьих, внедрение системы управления контентом открывает множество новых возможностей: проведение опросов, создание статей, публикация книжек, — и все они достаточно просты в освоении. Что тут главное? Главное — не запутаться в различных возможностях и составить план.

Чтобы в темноте не заблудиться,
Чтобы никогда с пути не сбиться,
Чтобы в нужном месте приземлиться, приводниться, —
Начерти на карте план.

В.С. Высоцкий

План, конечно, зависит от ваших текущих задач. В нашем случае получился следующий план:

1. **Перенести существующий сайт на Drupal**
   - собрать информацию и описать ее в виде кодов;
   - ввести информацию в Drupal;
   - описать новую систему навигации;
   - настроить внешний вид.

   Результатом будет сайт, который станет аналогом первоначального варианта.

2. **Добавить новые возможности**
   - выбрать новые возможности из списка и ранжировать их по порядку;
   - описать, как мы видим использование каждой новой возможности;
   - внедрить эти возможности по очереди.

   Результатом станет обновленный сайт с новыми возможностями для помещения и обработки информации.

3. Масштабировать сайт:
   - перевести сайт на другой язык (английский для начала);
   - сделать отдельный домен для англоязычного сайта (com);
   - ввести еще один язык (немецкий?).

   Результатом должен быть русский сайт на домене .ru, а оба других — на домене .com.

4. **Организовать сообщество**

Сейчас каждый второй сайт — это сообщество. Сообщество чародеек. Сообщество «Дома 2». Сообщество седьмой воды на киселе. Скоро сообществ будет больше, чем пользователей. Но, может быть, это правильно, и коллективный разум победит.

Для чего нам нужно сообщество:

- есть основная тематика сайта — мы хотим получать отклики и комментарии пользователей по ней, чтобы учесть новые идеи и развивать сервис;
- мы готовы публиковать статьи по своей тематике о рекомендациях и успешных случаях использования сервиса;
- мы можем поддерживать и развивать побочные тематики, связанные с нашей деятельностью, — это может быть интересно не только нам.

Результатом будет создание сайта сообщества как для пользователей сервиса, так и для людей, которые интересуются одной из тематик.

Этот план мы и попробуем реализовать шаг за шагом, описывая в данной книге, что у нас получилось.

ВЕРСИИ DRUPAL

- Обзор версий
- Нововведения в Drupal 7
- Важные предупреждения

Какую же версию системы использовать? Версий Drupal, как оказалось, великое множество. Реальные три:

- **Drupal 5** — старая версия Drupal. Используется на множестве сайтов, но нет никакого смысла ее изучать, поскольку версия 6 уже полноценно работает, а версия 7 находится на финальной стадии разработки;
- **Drupal 6** — текущая промышленная версия, готова к использованию на коммерческих сайтах. При необходимости создать сайт, который должен быстро и надежно заработать, нужно использовать ее;
- **Drupal 7** — версия в разработке, которая обещает много нового, но пока содержит достаточно большое число ошибок. Я надеюсь, что на момент издания книги мы уже будем иметь стабильную версию. Ее-то и нужно использовать начинающим разработчикам, поскольку именно эта версия будет активно использоваться в ближайшем будущем.

Данная книга посвящается Drupal 7. Мы вкратце опишем отличия Drupal 7 от Drupal 6 на случай, если кто-то из читателей уже имеет опыт работы со старой версией (если вы новичок, то просто пропустите этот список).

1. Полностью переделан интерфейс администрирования. Появилось верхнее меню, показ страниц администрирования поверх основных и возможность создания дополнительных строк с меню. Само меню полностью изменилось (первое время будет очень трудно, но вы быстро привыкнете).

2. Появился специальный отчет Dashboard, который также можно настраивать по своему желанию.
3. Очень сильно изменились темы: от стандарта HTML, который теперь потенциально поддерживает RDF (Resource Decision Framework), и названий регионов до списка используемых переменных, аргументов, функций и списка стилей. Это, пожалуй, самое серьезное изменение (полный список — по адресу: http://drupal.org/update/theme/6/7).
4. Добавилось множество полезных мелочей:
   - модуль CCK (Content Construction Kit) теперь встроен в ядро;
   - появились контекстные ссылки для элементов страницы (модуль Contextual Links);
   - улучшилось редактирование блоков;
   - стала гораздо удобнее форма редактирования контента;
   - появилось больше пояснений;
   - изменились региональные настройки;
   - сильно расширился список стандартных блоков. Рекомендуем пройтись по данному списку и понять, какие новые модели вам необходимы. При этом станут понятны и стандартные возможности нового ядра;
   - изменилось API доступа к базе данных — стало красивее и удобнее;
   - изменился формат .info для модулей и появился реестр кода, который хранит интерфейсы к модулям.

Сразу предупредим, что новичкам будет проще. Еще более важное предупреждение состоит в том, что за время пути... Имеется в виду, что Drupal 7 развивается, и к моменту получения окончательной версии могут появиться новые возможности (и самое главное — слегка измениться старые). Но от этого не застрахована ни одна развивающаяся система, поэтому будем терпеливы.

В данной книге мы будем выделять пункты меню и элементы интерфейса полужирным шрифтом, например **Конфигурация**. Тексты программ будут отображаться моноширинным шрифтом, например: `print $content`. Если ошибемся, то не карайте нас строго.

Ну, начнем...

ПЕРВЫЕ ШАГИ

- Используемая версия
- Вход в систему
- Проверка целостности
- Русификация
- Обзор меню
- Роли пользователей
- Краткий план работ

Не будем детально описывать, как нужно устанавливать Drupal. Отмечу только, что все подобные системы устанавливаются примерно одинаково и требуют веб-сервер Apache, интегрированный с ним PHP и MySQL или SQLite в качестве сервера баз данных (существует также возможность работы с PostGrep). Есть проект Денвер, который содержит все эти утилиты под Windows в одном пакете. При установке на Linux и прочие Unix нужен FTP для складывания файлов и, возможно, phpMyAdmin для MySQL.

Итак, мы будем считать, что вы установили Drupal с нуля. Версия, которую мы использовали:

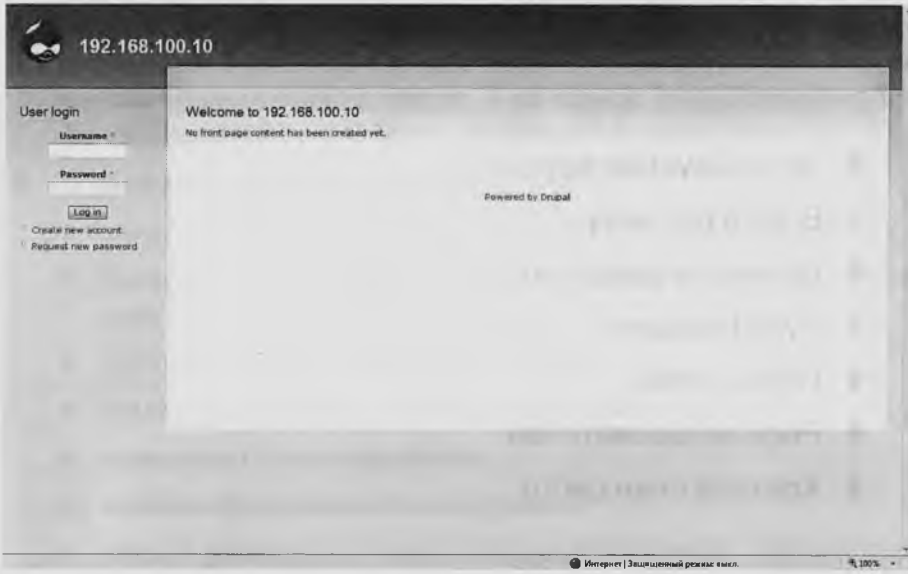
Drupal — 7.0-alpha5

MySQL — 5.0.45

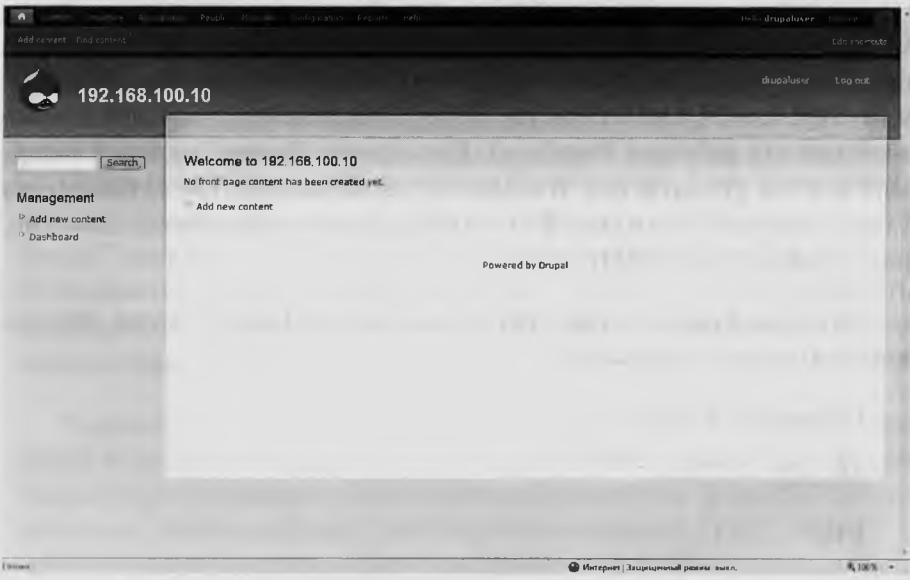
PHP — 5.3.1

Web server — Apache/2.2.14 (Unix) PHP/5.3.1

У вас должны быть имя и пароль администратора, которые создаются при установке. Наберем URL сайта, куда установили систему, и посмотрим, что нас ждет.



Вводим логин и пароль и входим в систему.

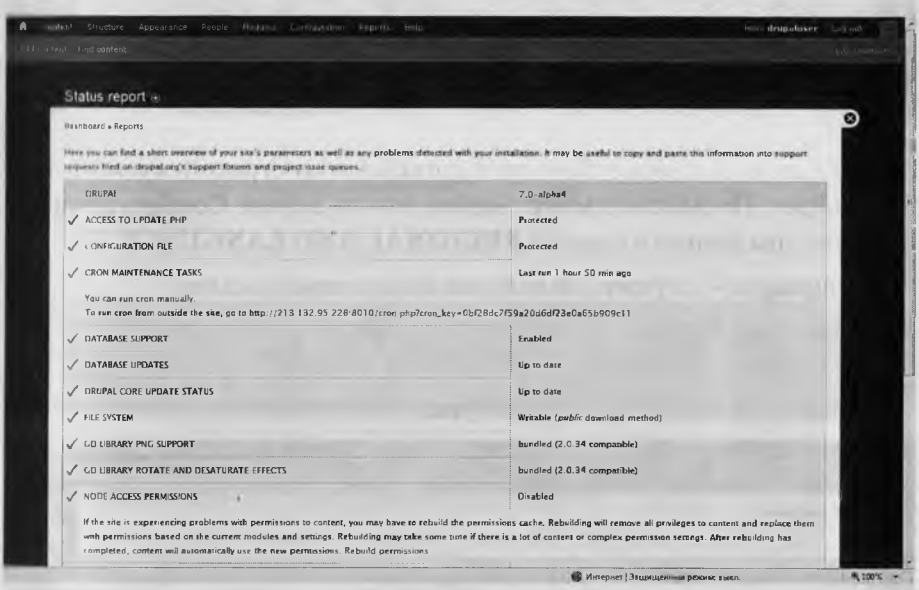


Ура! Мы в системе. Осмотримся. Сверху располагается основное меню администрирования. Под ним — полоса дополнительного быстрого меню. А дальше следует содержание страницы. Страницы пока нет, поэтому мы ничего не видим — просто Welcome.

Первое, что нужно сделать — это зайти в **Reports ⇒ Status report**. Там приводится краткий отчет по системе и список проблем.

Прежде чем использовать систему, необходимо добиться, чтобы для всех пунктов был показан статус **Ok** (зеленая галочка).

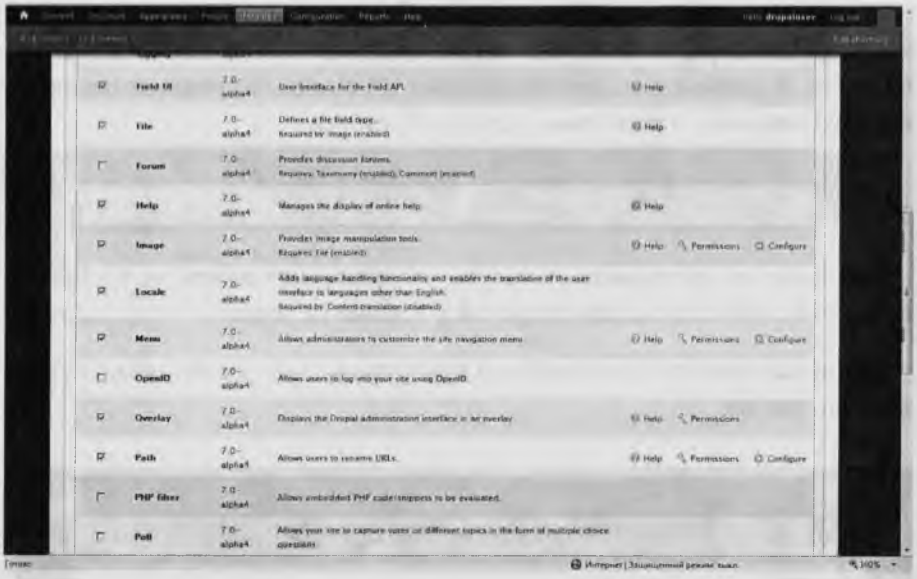
При наличии любых ошибок и предупреждений необходимо исправить ситуацию.



Полезно будет русифицировать интерфейс администратора. Иногда удобнее работать на своем языке, да и новичку разобраться в системе будет проще.

Кроме того, все, что относится к датам и времени, будет отображаться более привычно.

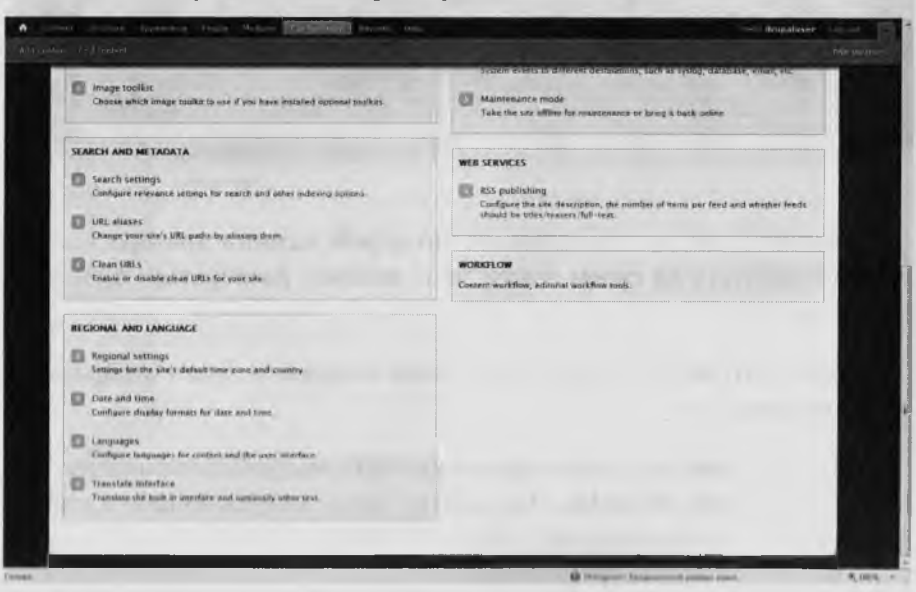
Для русификации необходимо включить модуль локализации. Делается это в меню **Modules**. Находим в списке модуль **Locale** и ставим флажок.



Щелкаем по кнопке **Save configuration** и со страхом ждем, что же произойдет. Порыскав в меню, видим, что на экране **Configuration** добавились два пункта в разделе **REGIONAL AND LANGUAGE**:

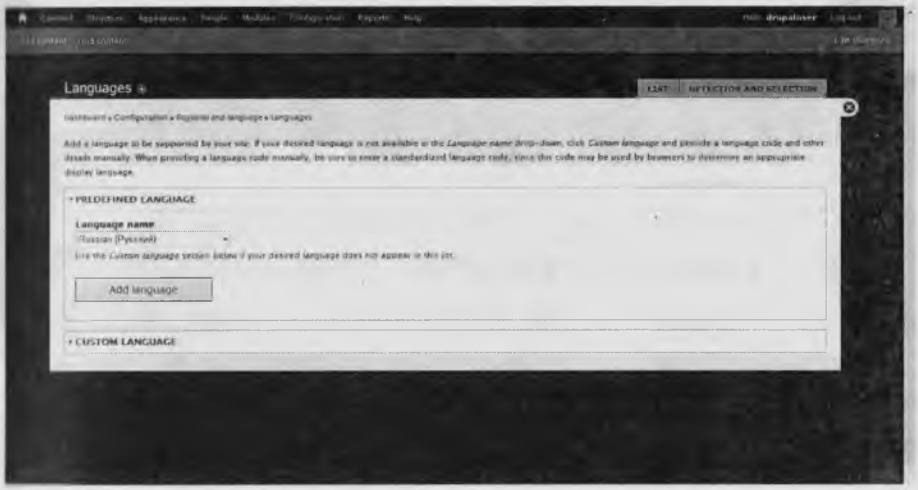
- **Languages** — служит для задания списка языков сайта;
- **Translate interface** — позволяет русифицировать интерфейс администратора, а потом и другие записи в интерфейсе.

Пока нам нужен явно второй пункт.



Есть два способа импортировать перевод. Сначала его надо скачать с сайта drupal.org. Там есть пункт **Translations** — ищите в нем. Мой файл назывался drupal-7.0-alpha4-ru-translations.tgz. Правда, в тот момент его можно было получить только с сайта drupaler.ru.

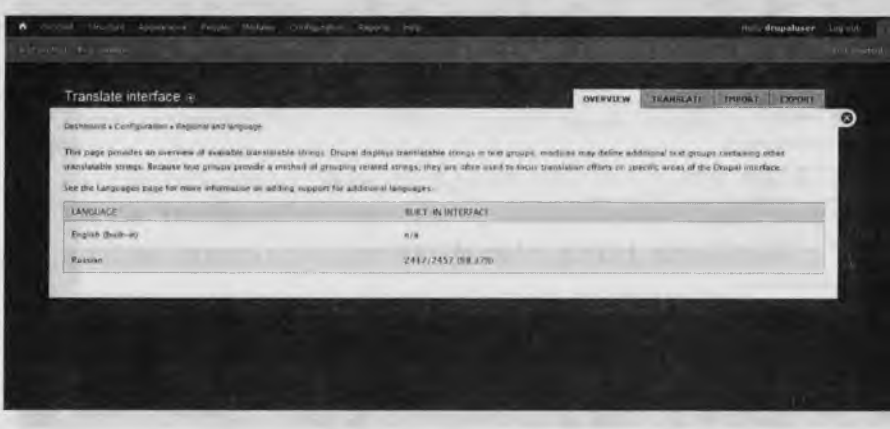
Первый способ — распаковать файл в основную директорию, где стоит Drupal, и добавить язык в меню (**Configuration ⇒ Regional and languages ⇒ Languages ⇒ Add Language**). При этом обещают импортировать все файлы языка, которые были записаны. Это файлы с расширением .po.



Потом ставим русский язык по умолчанию.

Второй способ — зайти в меню **Configuration ⇒ Regional and la guages ⇒ Translate interface ⇒ Import** и последовательно добавить в файлы .po, которые вы можете распаковать на клиентской машине. Э достаточно трудоемко, но зато вы примерно поймете назначение кажд го файла. Основной перевод находится в модуле **system**. Далее, импо тируя различные модули, вы увидите, как меню сайта и все части пот хоньку начнут переводиться на русский язык (то есть сайт представля собой множество кусочков, и иногда непросто определить, какой кус чек откуда берется). Однако лучше используйте первый метод.

Не совсем понятно, что за цифры и процент перевода выдается в о чете после импорта модуля. Цифры там чрезвычайно загадочные.

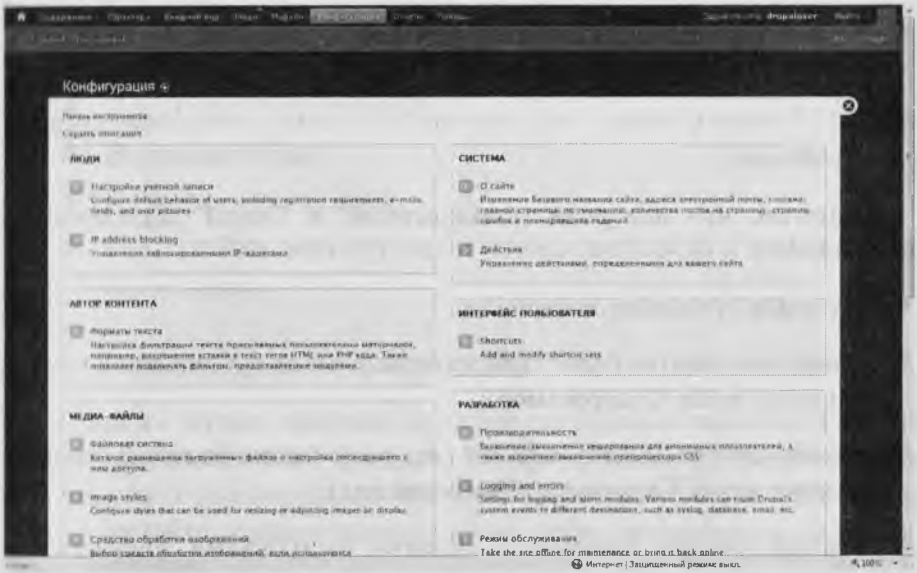


Общий результат — **2417/2457 (98,37%)**. Ну и ладно. Хорошо, чт не 100,38%.

После русификации посмотрим, что есть в меню:

- **Содержимое** — служит для работы с контентом; здесь можно вво дить и редактировать материалы;
- **Структура** — служит для описания способа отображения контент в виде сайта;
- **Внешний вид** — работа с темами показа содержимого;
- **Люди** — работа с пользователями и их правами;
- **Модули** — список установленных модулей и их включение/отклю чение;
- **Отчеты** — различные отчеты о работе и настройках;

- **Помощь** — помощь по модулям;
- **Конфигурация** (разберем подробнее) — это основной пункт (далее — просто **К ⇒**).



В принципе комментарии тут и так достаточно развернуты, но все же:

- **К ⇒ Люди** — работа с общими настройками для пользователей. Можно также описывать дополнительные поля для пользователей и управлять их отображением. В эту группу зачем-то включили и блокировку IP-адресов;
- **К ⇒ Система** — общие параметры сайта и работа со сложными действиями;
- **К ⇒ Автор контента** — настройка фильтров для текста вводимых авторами материалов;
- **К ⇒ Интерфейс пользователя** — здесь настраиваются и изменяются быстрые меню. Та же самая функция доступна при щелчке по кнопке **Edit shortcuts** (Редактирование быстрых меню);
- **К ⇒ Медиафайлы** — опции для расположения файлов и конвертации картинок при загрузке;
- **К ⇒ Разработка** — опции для администрирования работы сайта;

- **К ⇒ Поиск и метаданные** — настройки поиска и опций, влияющ на адреса страниц;
- **К ⇒ Web-сервисы** — настройка взаимодействия с различными се висами, в частности, с RSS;
- **К ⇒ Процесс** — настройка различных опций документооборо Пока здесь пункты отсутствуют;
- **К ⇒ Регион и язык** — это, конечно, все, связанное с языковыми н стройками.

Заметим, что все роли пользователей в Drupal разделены смыслу:

- авторы могут писать материалы;
- редакторы заняты обработкой материалов и их публикацией на са те (пункт меню **Содержимое**);
- дизайнеры описывают внешний вид, размещение блоков и структ ру навигации (**Структура** и **Внешний вид**);
- администраторы занимаются общими настройками, управлени пользователями, регулярным мониторингом отчетов, обновлениям системы и установкой новых модулей (все остальные пункты);
- разработчики порождают новые модули и настраивают внешн вид, а также другие функции существующих модулей (а им вооб интерфейс практически не нужен — они работают в текстовом р дакторе).

Однако пора заняться сайтом. Ранее при входе в Drupal 6 на главн странице вам предлагали план:

- настройте свой сайт;
- добавьте модули;
- настройте дизайн;
- введите материалы.

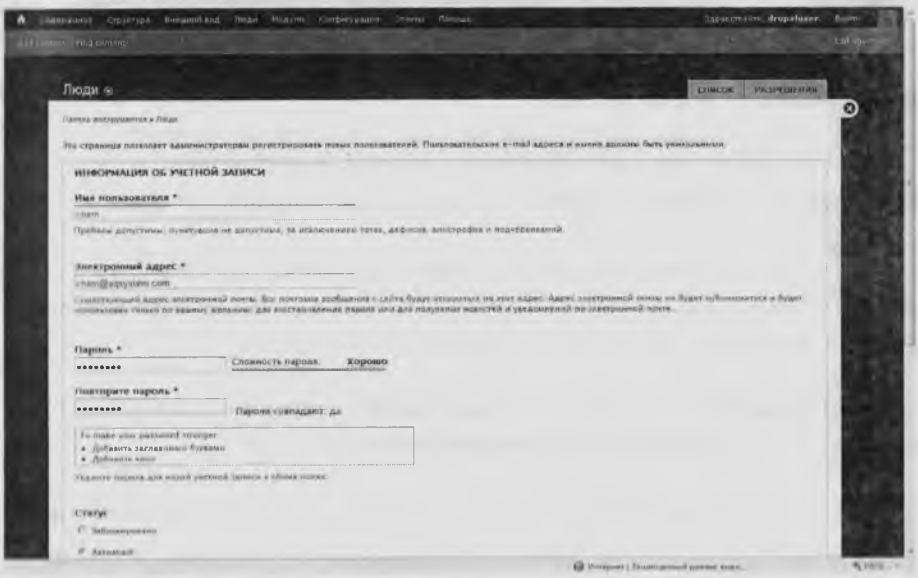
Сейчас в Drupal 7 такого плана пока нет, но, может быть, он появи ся в будущем. Из этого плана следует только одно — перед тем как н чать выполнение нашего плана, нужно выполнить ряд общих настро и установить дополнительные полезные модули. Давайте этим и за мемся.

ПОЛЬЗОВАТЕЛИ И ПРАВА

- Создание пользователей
- Роли пользователей
- Задание прав
- Присвоение ролей
- Настройка регистрации пользователей и входа в систему

Одна из первых настроек, которые необходимо выполнить, — это создание пользователей и раздача им прав. Работа от имени главного администратора возможна, и он по умолчанию всесилен, но это моветон. Предпочитайте всегда создавать специального пользователя, даже если работаете один.

Создание пользователя происходит через меню **Люди**. Выбираем **Добавить пользователя**, вводим свои данные и щелкаем по кнопке **Регистрация**.



Выходим из системы как администратор и вводим свое имя и пароль.

Входим и понимаем, что не можем вообще ничего. Мы видим просто сайт только с собственным именем в правом углу.

Чтобы работать с системой полноценно, нам нужно обеспечить себе права доступа.

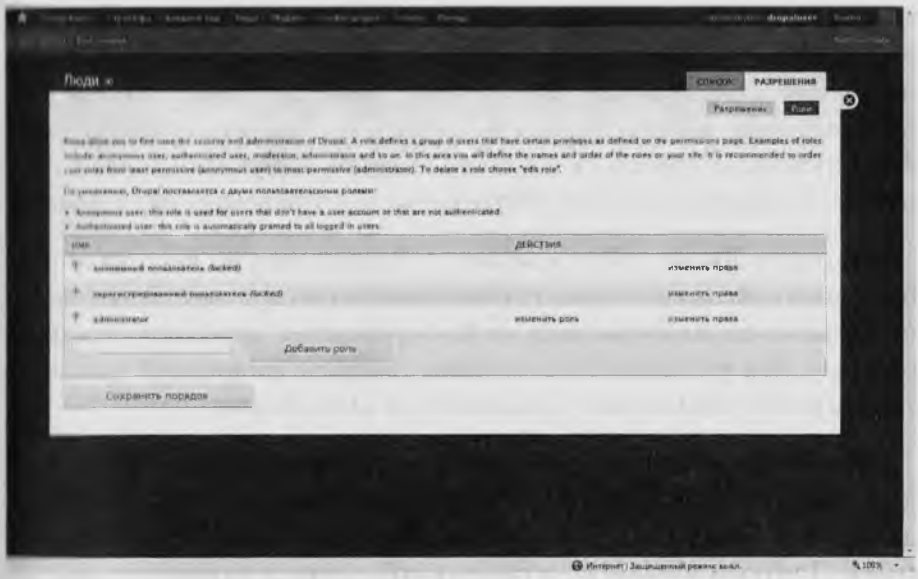
В системе Drupal пользователь, который создан при инсталляции, всегда обладает всеми правами.

Так решается вопрос о возможности удаления последнего администратора, ну и, наверное, проблемы при задании прав на модули.

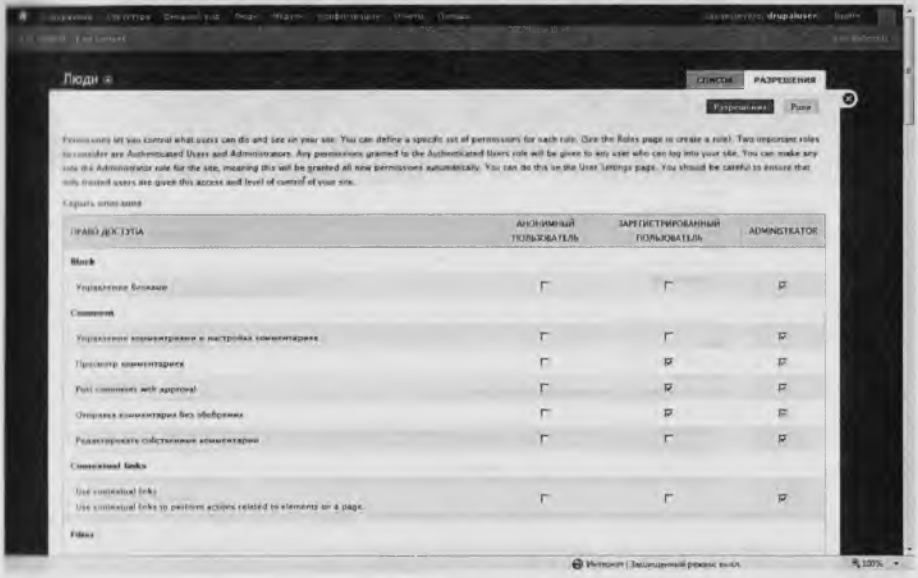
Но если мы хотим передать все права другому пользователю, сначала необходимо создать роль, потом отдать ей все права, а уже после этого указать, что новый пользователь выполняет эту роль.

Снова входим в систему от имени главного администратора и следуем в меню **A** ⇒ **Управление пользователями** ⇒ **Роли**.

Ранее нам приходилось создавать специальную роль **administrator**, теперь, слава богу, все сделано до нас.



Назначение прав для ролей осуществляется в меню **Люди** ⇒ **Разрешения** ⇒ **Разрешения** (почему данный пункт не был переведен как «Права», не очень понятно — видимо, специалисты предпочли прямой перевод с английского «Permissions»).



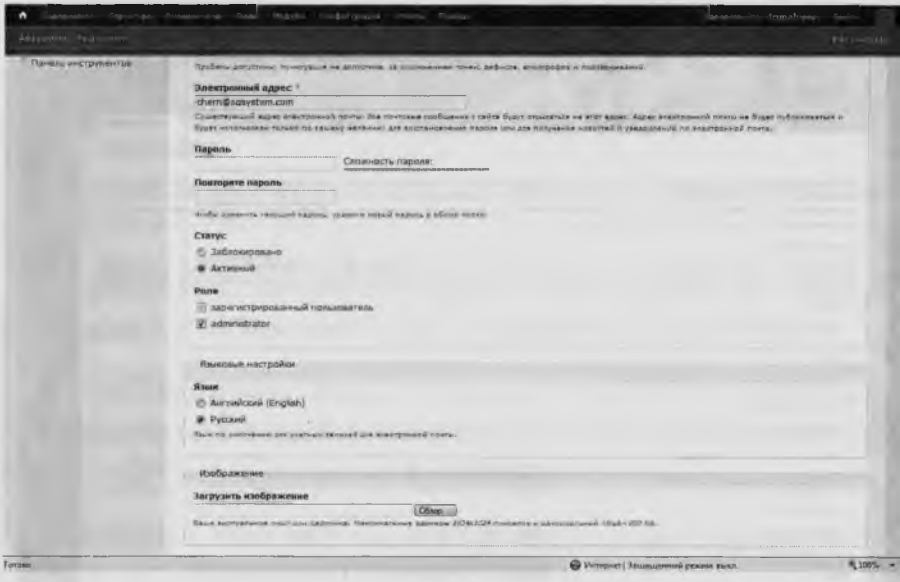
Мы позволим себе некоторое отступление.

Весь Drupal, в том числе и функционал, относящийся к администри рованию, состоит из модулей. Полное единообразие. При установк присутствуют 35 модулей, 19 из которых по умолчанию включены (м дополнительно включили модуль **Locale**). Все эти модули составляю ядро системы. Перейдя на страницу **Модули**, мы увидим список все инсталлированных модулей. Включенные модули помечены галочкой.

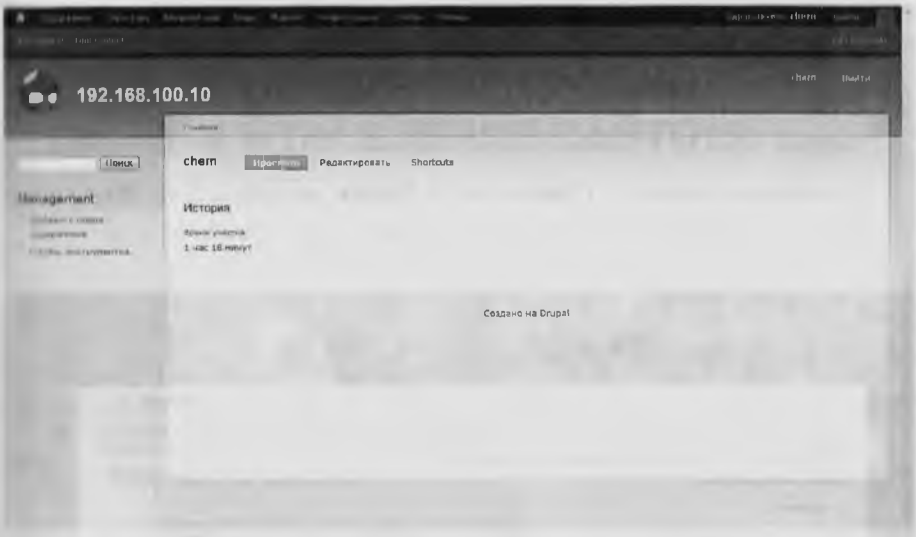
В пункте **Люди** ⇒ **Разрешения** ⇒ **Разрешения** по вертикали ото бражается список всех прав, сгруппированных по модулям. Здесь указа ны только подключенные модули. По горизонтали приводятся все рол системы. При установке были созданы три роли: анонимный пользова тель, зарегистрированный пользователь и administrator.

Мы можем создать еще несколько ролей (например, для автора, ре дактора и т. д.). Тогда при подключении любого нового модуля необхо димо будет заходить на данную страницу и добавлять галочки для эти ролей. Для роли administrator галочки добавляются автоматически.

Теперь нужно не забыть присвоить себе новую роль. Переходи в опцию **Люди**, находим себя в списке, указываем **Изменить**, стави флажок возле роли administrator и выполняем сохранение.

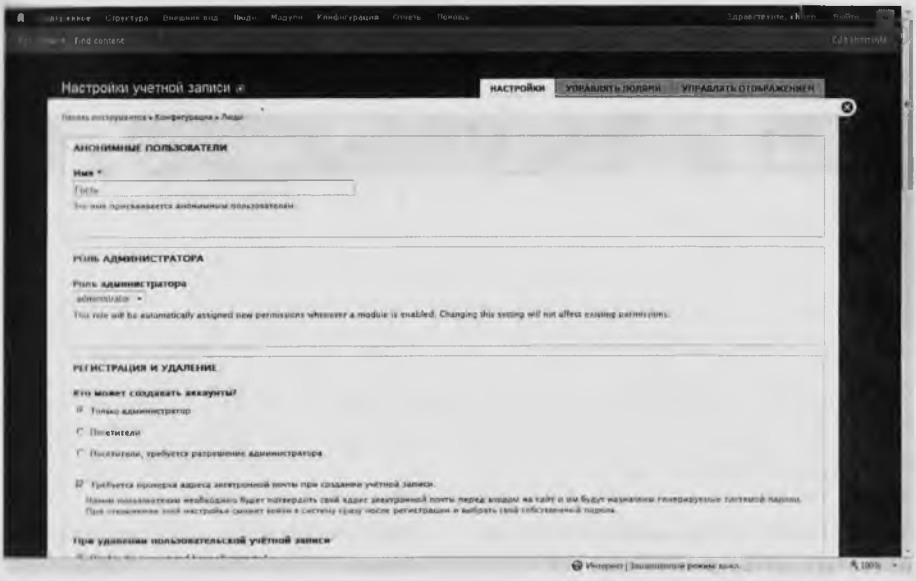


Теперь можно зайти под собственным логином и проверить, отобра жается ли полное меню.



Для реализации нашего плана необходимо выполнить еще пару полезных вещей:

1. Устанавливаем, что создавать пользователей может только администратор. Данная опция не понадобится до тех пор, пока мы не соберемся создавать сообщество, позволив регистрацию других пользователей. Переходим в пункт **К** ⇒ **Люди** ⇒ **Настройки** учетной записи и отмечаем радиокнопку **Только администратор** в группе **Кто может создавать аккаунты?**. Сохраняем изменения.



После этого из блока ввода логина и пароля исчезнет кнопка **Регистрация**.

2. Отключаем блок логина с центральной страницы. Это нужно, чтобы данный блок не мешал нам при оформлении собственного сайта.

   Выполняем команду **Структура ⇒ Блоки**, находим блок **Вход в систему** и указываем, что область для него отсутствует.

Щелкаем по кнопке **Сохранить блоки**, выходим и проверяем результат.

И сразу же начинаются проблемы: «А как я зайду на сайт»? Необходимо использовать ссылку http://адрес_сайта/user. Добавьте ее в папку Избранное.

Пришло время заняться общими настройками и поэкспериментировать с их возможностями.

ОБЩИЕ НАСТРОЙКИ И УПРАВЛЕНИЕ САЙТОМ

- Настройки учетной записи
- О сайте
- Форматы текста
- Shortcuts (Быстрые меню)
- Файловая система
- Средство обработки изображений
- Производительность
- Logging and errors (Логи и ошибки)
- Сайт на обслуживании
- Региональные установки
- Дата и время
- Внешний вид

Общие настройки распределены по группам в меню **Конфигурация**. Мы будем изучать настройки в порядке их расположения и разберем только составляющие, которые имеет смысл откорректировать сразу. Остальные будем потихоньку подстраивать в процессе дальнейшей работы. Поскольку настройки регулярно добавляются, то может показаться, что это процесс бесконечный. Но не отчаивайтесь — немного практики, и вы легко будете знать, где и что надо подправить, чтобы достичь цирваны.

Настройки учетной записи

Здесь приведены настройки, которые относятся к пользователям. Дело в том, что в Drupal 7 пользователь практически приравнен к материалу. Мы можем задавать дополнительные поля для любого пользователя, и при отображении его страницы они будут показаны. Но пока не будем обращать внимания на поля, а займемся непосредственно настройками.

Настроек тут достаточно много. В принципе опции по умолчанию нас устраивают. Надо лишь убедиться, что **Роль администратора** установлена в administrator. Это избавит нас от необходимости добавлять флажки при появлении новых модулей. С остальными опциями вы можете поэкспериментировать самостоятельно.

О сайте

Вот это самый полезный пункт. Здесь можно развернуться. Измените, например, название сайта (ну вот, вы уже практически создали свой сайт). Остальные поля тоже можно заполнить, но это необязательно. Такой способ не является основным, просто он дает возможность быстро получить некий видимый результат. Впрочем, введите электронный адрес. Он часто отличается от адреса администратора. Заодно разбери-

тесь, через какой SMTP-сервер идет отправка, и в каком домене нужно располагать адрес.

Всем известно, что при щелчке по линку, для которого нет страницы, выводится всякая ерунда и ошибка 404. А если создатели сайта хотят показаться «крутыми», они отправляют пользователя на страницу с извинениями и советами. Уровень советов целиком зависит от вашей фантазии. Здесь же можно настроить специальные страницы для ошибок 403 и 404. Это не так важно на начальной стадии, но необходимо для корректной раскрутки сайта.

Кстати, упомянем еще один очень интересный момент. Если заполнить данные о сайте, сохранить их и потом закрыть окошко администрирования, на сайте ничего не изменится, потому что по умолчанию окно администрирования работает в режиме оверлея, никак не задевая основное содержимое. Чтобы увидеть изменения, нужно нажать клавишу **F5** (Обновить). Так происходит всегда, поэтому прежде чем искать причину того, почему изменения не отобразились, обновите страницу. Если это не помогло, сотрите кэш и обновите страницу еще раз. Про кэш мы поговорим далее в этой главе.

Если режим оверлея вам не нравится, можно отключить его, открыв меню **Модули** и отключив модуль **Overlay**.

Форматы текста

Некоторые непорядочные пользователи могут попытаться взлома
сайт, введя специальные HTML-конструкции. Когда эти конструкц
отобразятся на странице, они либо сломают красивую картинку сай
либо (что еще хуже) начнут рассылать вирусы по самому сайту и кл
ентским компьютерам.

Чтобы этого избежать, необходимо проверять сообщения от так
пользователей и выкидывать все лишние конструкции. Программ
проверяльщики называются *фильтрами*, и через эти фильтры проход
все вводимые сообщения пользователей. Набор и тип фильтров завис
от роли пользователя.

Тут нас все устраивает. При наличии большего числа ролей в да
ном меню можно создать другие типы фильтров и детально задать ра
решенные к вводу HTML-теги.

Shortcuts (быстрые меню)

Здесь настраиваются дополнительные меню для быстрого доступ
Можно создать несколько полосок, в каждой — до семи пунктов мен
Рассмотрим окно добавления пункта.

Путь указывается в виде URL. То есть, чтобы добавить пункт меню, сначала нужно зайти на страницу, которую мы хотим добавить, и посмотреть ее URL.

Например, для страницы **Конфигурация** URL выглядит так: http://имя сайта/user/5#overlay=admin/config.

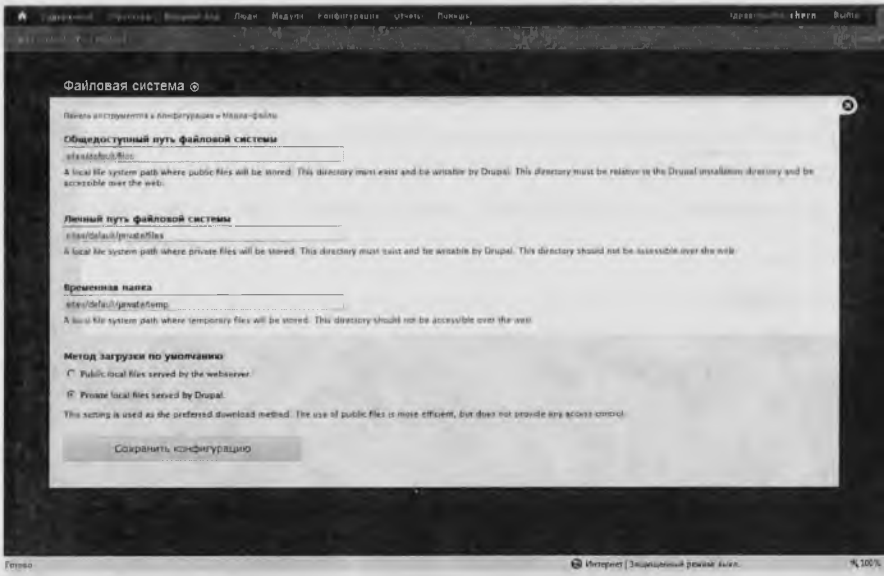
admin/config и есть URL, который надо вписать в путь. При отсутствии оверлея мы просто попадем на эту страницу без страницы /user/5 вверху. Достаточно неудобно.

Но есть более простой способ. Посмотрите на плюс справа от заголовка странички **Конфигурация**. Щелкнув по нему, мы как раз и встанем текущую страницу в быстрое меню.

Файловая система

Обычно тут не задан **Личный путь файловой системы**. В Drupal 6 традиционно по умолчанию он указывался в sites/default/private/files.

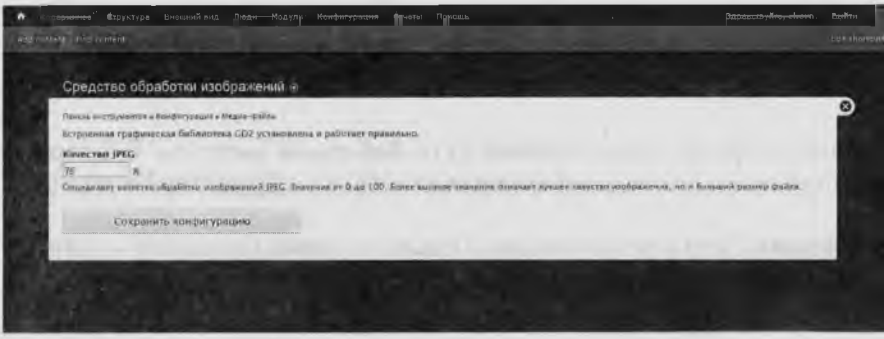
В версии 7 путь не изменился. При этом может возникнуть ошибка, указывающая на невозможность создать данный каталог — тогда нужно проверить права на директорию sites/default.



После сохранения можно будет выбрать **Метод загрузки по умо**
чанию. Поскольку мы делаем сайт для опубликования общей информ
ции, указываем вариант **Public**.

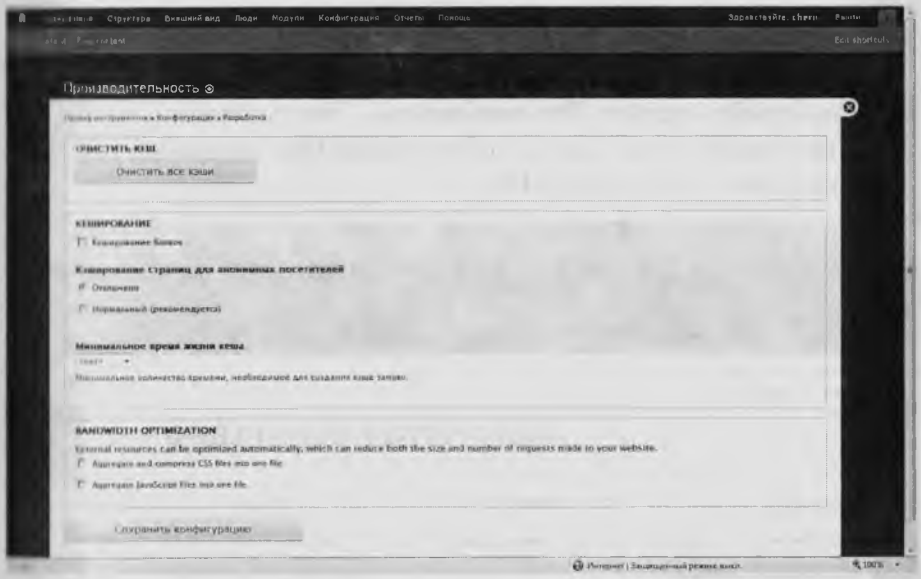
Средство обработки изображений

Здесь можно убедиться, что стандартная графическая библиоте
установлена и работает, а также задать стандартный уровень качест
при обработке изображений. Стандартное средство позволяет устан
вить максимальный размер картинки и сжимать изображения, котор
больше заданного, до максимально допустимого. При этом будет и
пользоваться данный уровень качества.

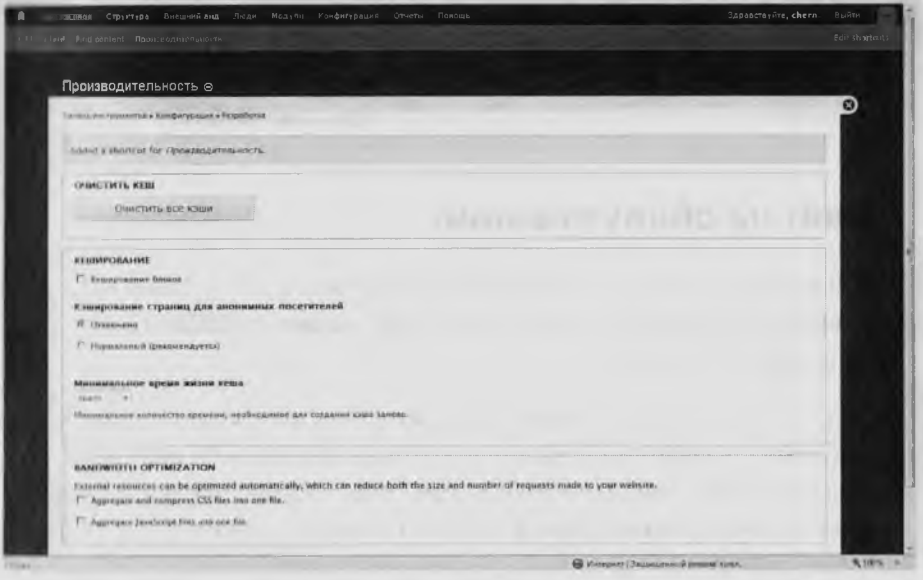


Производительность

Данный пункт понадобится при переводе нашего сайта в рабочий режим. Пока убедимся, что все кэши и оптимизации отключены — это может помешать нам при разработке.



Именно в данном пункте придется в дальнейшем регулярно очищать кэш. Добавим его сразу в быстрое меню.



Logging and errors

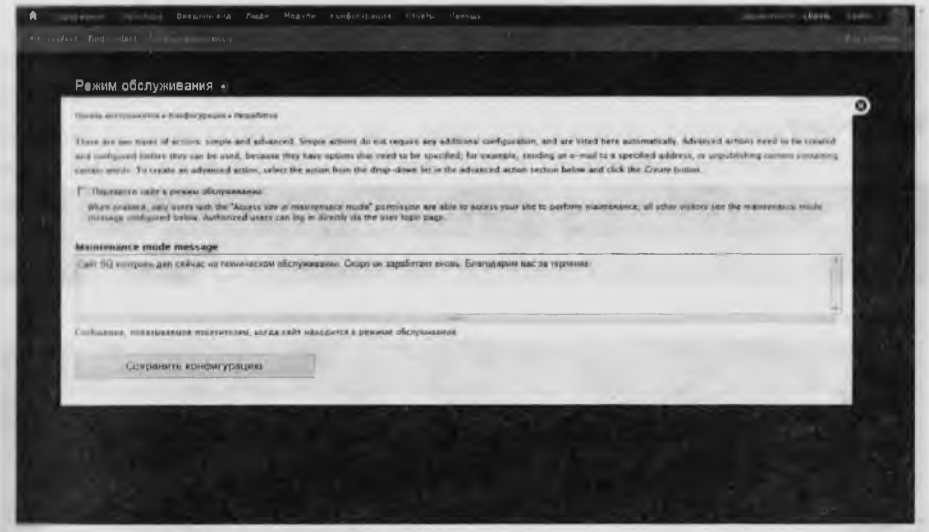
Здесь задается параметр сборки мусора. Он может быть важе
если вы пользуетесь shared-хостингом, чтобы не засорять место. Ес
у вас выделенный сервер, можно указать большее значение, наприм
10000.

Также полезна опция, управляющая выводом ошибок. При разр
ботке выбирайте вариант **Все сообщения** — всегда полезно, чтобы с
стема как можно чаще кричала об ошибках. При настройке рабоче
сайта, конечно, лучше указать **Нет**.

Сайт на обслуживании

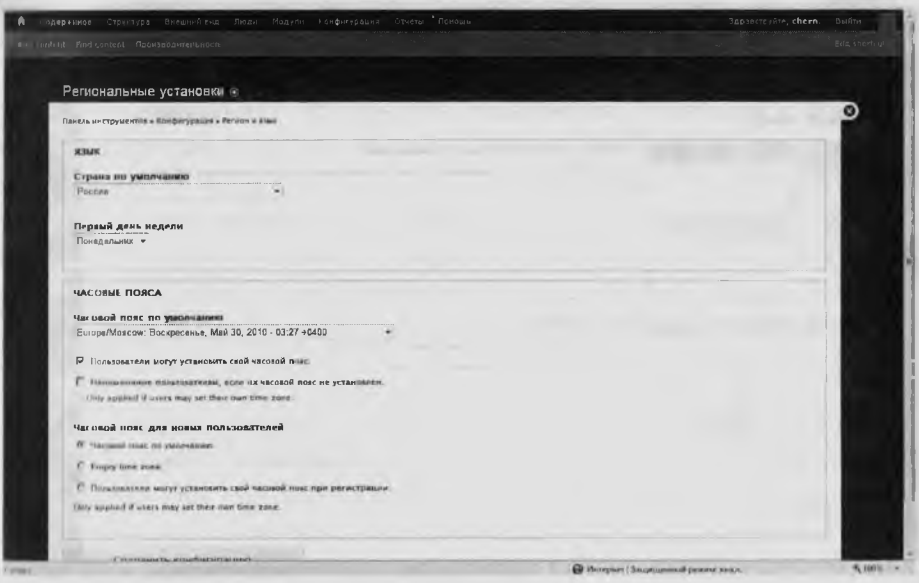
Пункт нужен, если есть желание остановить работу сайта на нек
торое время (например, при глобальной переделке стартовой страниц
чтобы ее никто не видел).

Мы переводим сайт в режим обслуживания, и все пользователи в
дят эмблему с названием и надпись, которая приведена в окошке. А с
стема забрасывает вас записями о том, что сайт находится в режим
обслуживания. Однако прямо из этой строки его можно вернуть в он
лайновый режим.



Региональные установки

Здесь надо указать параметры **Страна** ⇒ **Россия** и **Первый день недели** ⇒ **Понедельник**. Проверьте, правильно ли установлен часовой пояс. Остальные опции можно будет установить позже по своему усмотрению. Мы оставили пользовательские пояса — мало ли, откуда к нам заглянут пользователи.



Дата и время

Тут все понятно — форматы даты и времени. Укажем все установки для России.

Мы можем также определять собственные форматы дат и использовать их при необходимости.

Внешний вид

Здесь отображаются имеющиеся темы для показа данных.

По умолчанию включены две темы: **Garland** и **Seven**. Также есть одна отключенная тема **Stark**. Тема определяет оформление страниц при показе. Об этом мы, конечно, поговорим позднее. Пока надо усвоить одно — для страниц нашего сайта необходима одна тема. Это тема, которая в основном списке (с картинками) помечена как **тема по умолчанию**. После установки это тема **Garland**. Создав свою тему и добавив ее в список, мы определим ее в качестве темы по умолчанию. А пока давайте прокрутим экран вниз и посмотрим на раздел **Тема админки**.

По умолчанию тут установлена тема **Seven**. Поиграйте с темами и посмотрите, как будут меняться окна администрирования при выборе того или иного варианта. Не следует ставить только **Тема по умолчанию**, потому что при разработке собственная тема, как правило, сначала несколько ущербна, и администрировать в ней нельзя. Мы будем продолжать разработку в теме **Seven**, чтобы вас не путать. Вы же можете выбрать любую тему, которая вам по душе (в принципе они все очень похожи).

Весьма интересен второй флажок. Тема админки действует только на меню **Администрирование**. Все остальные пункты меню будут показаны в основной теме сайта. Второй флажок подключает тему админки при вводе материалов. Пока пользователи у нас только «свои», лучше установить данный флажок для простоты.

Потом, закончив свое оформление и подключив внешних пользователей, мы перенесем администрирование и ввод материалов в созданную тему. Тогда мы не удивим внешних пользователей переключением в другую тему.

Мы прошлись практически по всему интерфейсу. Остальные пункты будут разобраны, если они нам понадобятся. По крайней мере, мы знаем, где их искать. Надеюсь, дальше будет интереснее.

Инсталляция новых модулей будет добавлять пункты в данную секцию, так что ждите ее разрастания со временем. Когда модулей много, можно использовать модуль **Administration menu**, который принципиально меняет работу с меню и снимает большинство проблем.

СТАНДАРТНЫЕ И ДОПОЛНИТЕЛЬНЫЕ МОДУЛИ

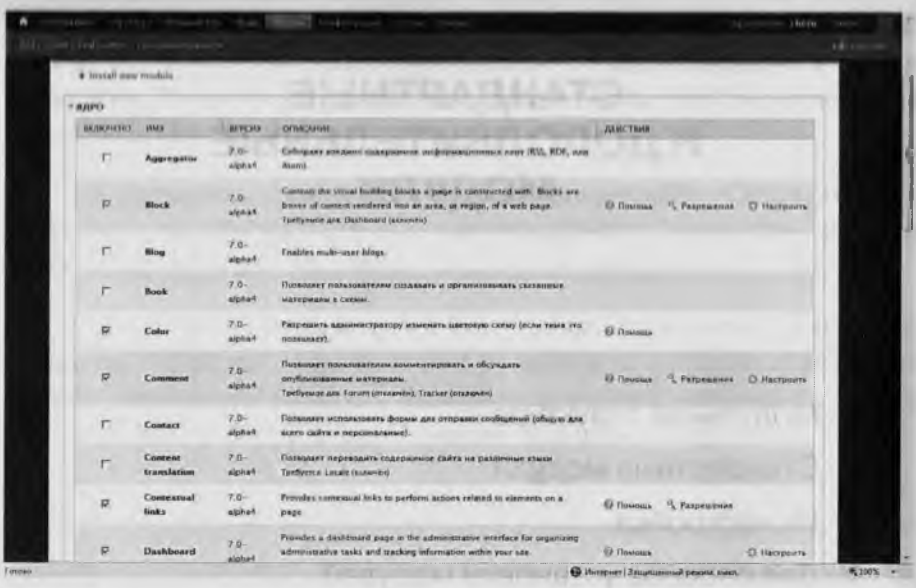
- Модульная структура
- Стандартные модули
- Их настройка
- Два способа установки модулей
- Administration menu
- Настройка нового меню

Как мы говорили, весь Drupal состоит из модулей. Желая создать форум, блог или любой другой сайт, мы должны собрать его из соответствующих элементов. Таких кусочков достаточно много. Некоторые из них лучше, другие хуже. При этом кусочки активно взаимодействуют друг с другом.

В общем, возникает дерево модулей (примерно, как технологии в игре «Цивилизация», если кто знает). Модули обычно добавляют составляющие в меню.

Существуют базовые модули (как умение делать горшки — без него никуда), а есть сложные и громоздкие варианты (как изготовление ракетных кораблей), без которых можно и обойтись. Как же разобраться во всем этом?

Для начала мы разберемся со стандартными модулями. Открыв меню **Модули**, мы увидим их список.



Здесь приводится информация по каждому модулю, а также дается возможность включать и выключать их.

Развернутое описание по включенным модулям можно получить в пункте **Помощь**.

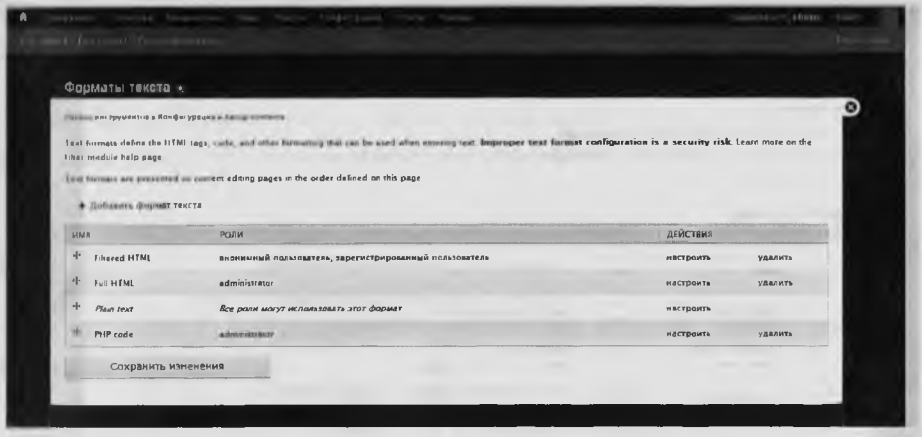
Знание модулей и умение ими пользоваться и есть основная задача администраторов и разработчиков сайта на Drupal. Набрали нужные модули, выбрали тему, подправили, и сайт готов.

У нас пока есть только стандартные модули. Не будем трогать все варианты, только включим еще стандартный модуль **PHP filter**, который позволяет включать PHP-код в материалы. Внимательно прочитайте инструкцию по нему!

Ставим флажок и щелкаем по кнопке **Сохранить**. Смотрим, что изменилось.

Можно почитать инструкцию по **PHP filter**. Это возможность использования формата данных.

Переходим в форматы данных (**К** ⇒ **Форматы текста**). Видим, что появился новый формат. Добавляем право на него нашей роли administrator.



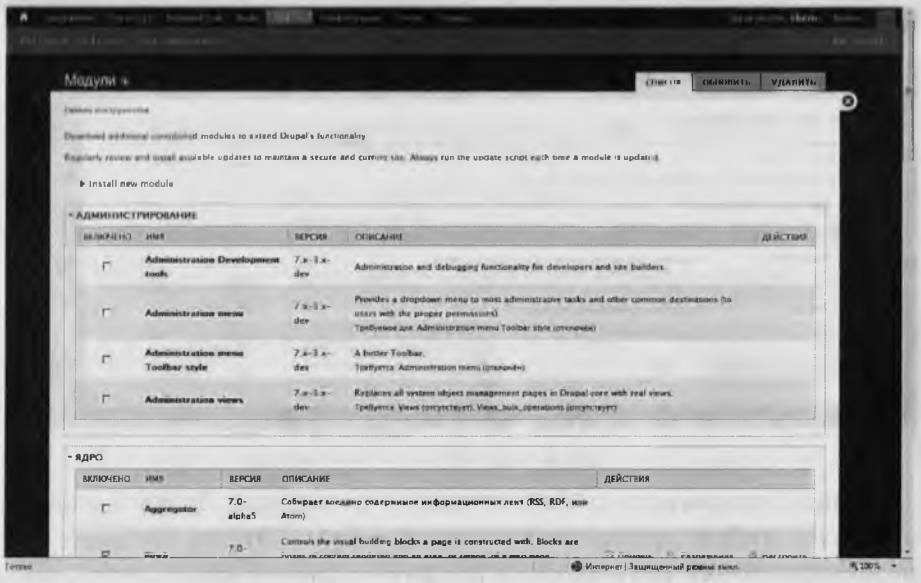
Теперь мы умеем включать и выключать модули. Давайте установим какой-нибудь принципиально новый модуль. Это то, что будет вам необходимо очень часто, поэтому процесс установки надо выучить наизусть.

В Drupal 7 есть два способа установки модулей:

1. Классический:
   - скачать модуль;
   - распаковать его в директорию drupal/sites/all/modules (не перепутайте с drupal/modules);
   - включить его в пункте **Модули**.
2. Новомодный:
   - в меню **Модули** щелкнуть по кнопке **Install new module** (Установить новый модуль);
   - ввести путь к архиву модуля (обычно с сайта drupal.org) и щелкнуть по кнопке **Установить**;
   - включить его в пункте **Модули**.

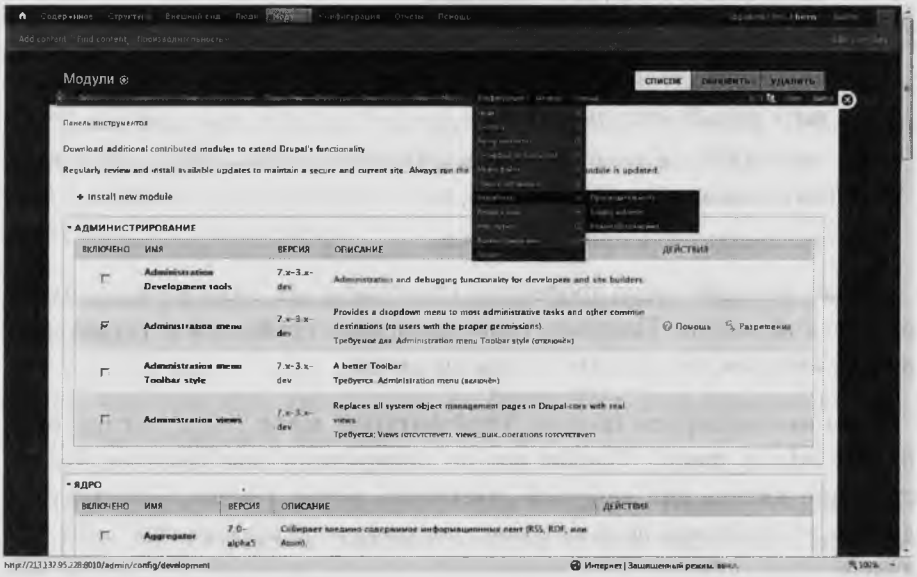
Классический метод проверен и точно работает. Новый метод может работать не всегда. Поэтому, если возникают проблемы с установкой новым методом, переходите на классический.

Мы инсталлируем модуль **Administration menu** (http://drupal.org/project/admin_menu). Сделав все по инструкции, мы заметим, что на странице включения модулей появилась новая группа. Данный модуль — это на самом деле не один, а целая группа взаимосвязанных модулей.



Давайте включим один из них — **Administration menu** — и посмотрим, что получилось.

Мы видим довольно странную картину:



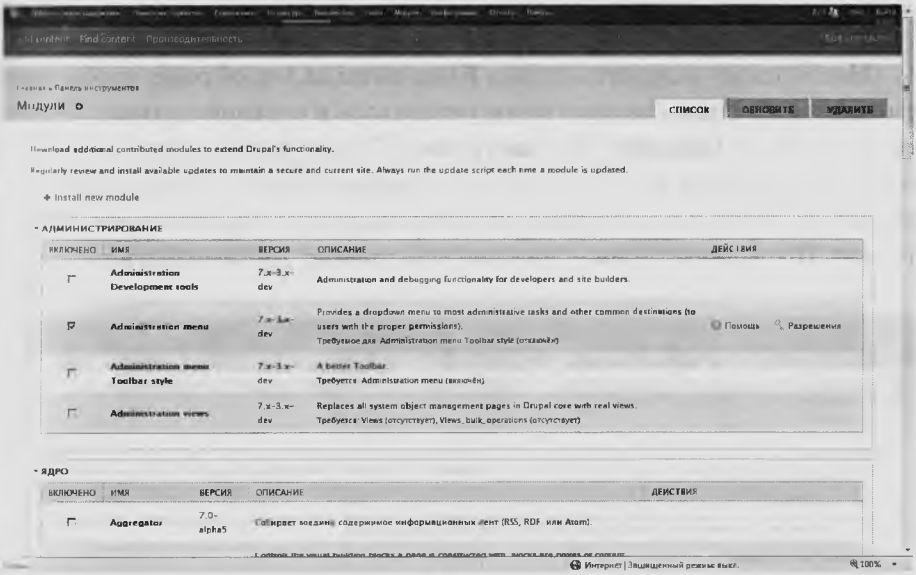
Теперь на экране отображаются два меню: первое — основное от Drupal 7, второе — от модуля **Administration menu**.

Они весьма похожи, но новое меню иерархическое, что сильно экономит как число щелчков, так и число обращений к сайту.

На самом деле развитие Drupal идет путем интеграции полезных функций из модулей в ядро.

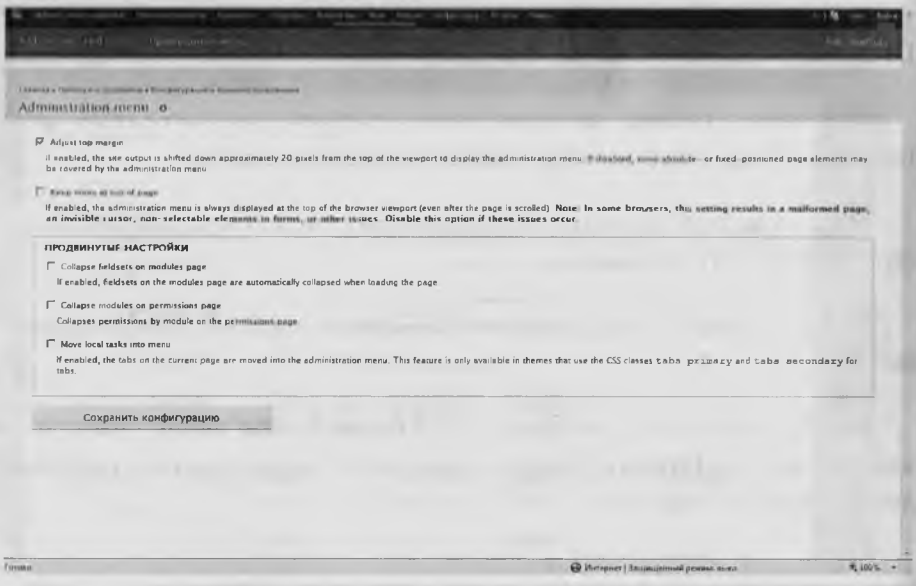
Меню Drupal 7 было сделано на основе модуля **Administration menu**.

Чтобы было удобно пользоваться **Administration menu**, нужно выключить модуль **Overlay**. При этом одно меню закроет другое, и мы увидим следующую картину.

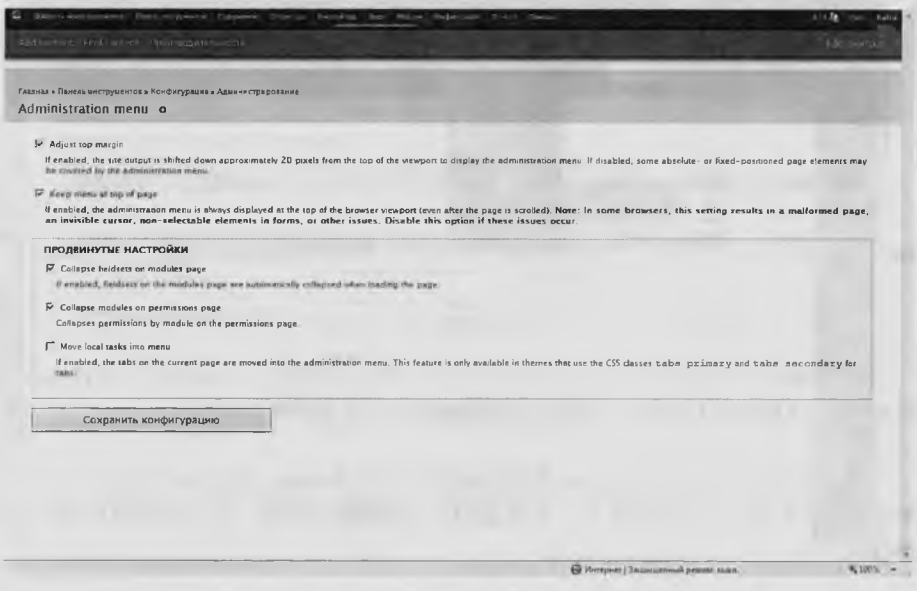


Закрывается оно не полностью, но это, видимо, можно подстроить стилями. Однако работе такое положение дел не мешает.

Заметим, что у модуля **Administration menu**, который мы включили, есть настройки в **К ⇒ Администрирование ⇒ Administration menu**.



Необходимо включить опцию **Keep menu at top of page**, чтобы при смещении по странице вниз меню оставалось в верхней строчке. Очень полезны также **Продвинутые настройки**. Только ради них уже стоит использовать данный модуль. Первые две опции позволяют группировать список модулей и разрешений, что весьма удобно. Включите их обязательно. Последняя опция — на ваше усмотрение. Должно быть так.



При написании книги во всех скриншотах мы используем базовое меню Drupal 7.

Потом вы можете включить иерархическое меню. Мы рекомендуем пользоваться именно им — это гораздо удобнее и быстрее.

Теперь, научившись добавлять модули, давайте скачаем и включим базовый полезный набор модулей, который нужен практически всем.

Все эти модули занимаются вопросами общего назначения, и мы разберем их функции в последующих главах.

КРАСИВЫЕ ССЫЛКИ И ВВОД МАТЕРИАЛОВ

- Чистые ссылки
- Token
- PathAuto
- Global Redirect
- Создание материала
- Настройки адресов
- Автоматические синонимы
- Настройки автоматических синонимов
- Transliteration

В Drupal есть понятие *чистые ссылки*. Что же это такое? Рассмотрим.

Drupal может использовать ссылки двух видов:

- www.example.com/?q=node/67;
- www.example.com/node/67.

Первый вид ссылки неудобочитаем и не очень хорошо влияет на работу поисковых серверов. Второй вид как раз и называется чистыми ссылками. Согласитесь, что они выглядят куда лучше.

В Drupal 7 функция чистых ссылок включена изначально. И выключать ее не надо. Мы просто добавим модули, помогающие обработке ссылок.

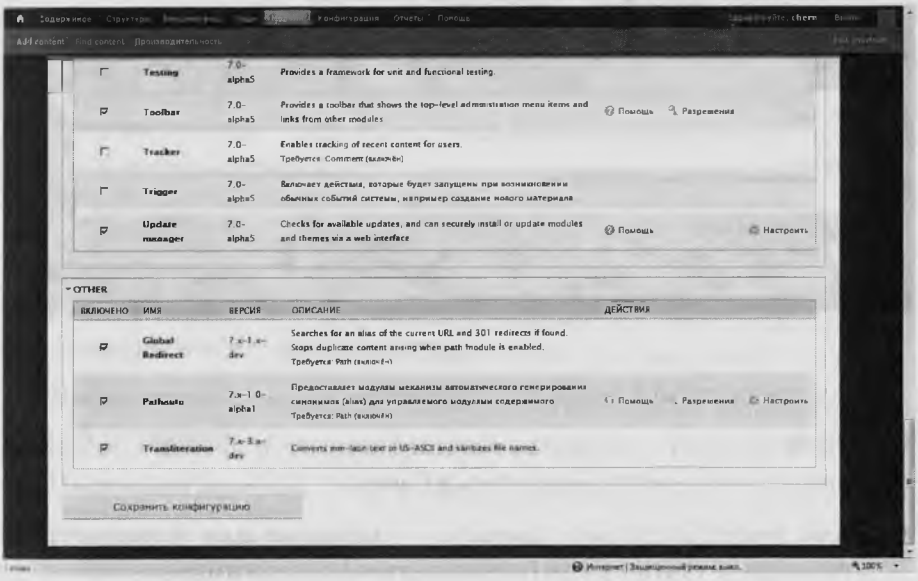
Первый модуль, который необходим, — это модуль **Path**. Он входит в ядро и нужен нам для создания синонимов страниц с материалами.

Дело в том, что в Drupal все базовые URL для материалов выглядят как node/id_node. А мы хотели перенести свой старый сайт с совершенно иными ссылками. Модуль **Path** позволяет создавать синонимы для URL страниц. При этом мы можем дать страницам те же URL, что и раньше.

Кроме модуля **Path**, необходимо установить еще три полезных модуля:

- **PathAuto** (http://drupal.org/project/pathauto) — позволяет автоматически создавать ссылки на основе правил. Это полезно, когда множество пользователей создают собственные страницы. На первом этапе мы можем самостоятельно ввести путь, но когда страниц будет много, пусть система формирует ссылки автоматически. Полезность таких ссылок состоит также в том, что они настраиваются в зависимости от типа материала;
- **Transliteration** (http://drupal.org/project/transliteration) — дает возможность автоматически переводить русские названия;
- **Global Redirect** (http://drupal.org/project/globalredirect) — позволяет включать уйму полезных защит, связанных с неверными вводами URL. Убирает лишний слэш в конце, обрабатывает ошибку 301, позволяет использовать любые буквы — строчные или прописные, и т. д.

Давайте активируем эти модули и посмотрим, что они нам дадут.



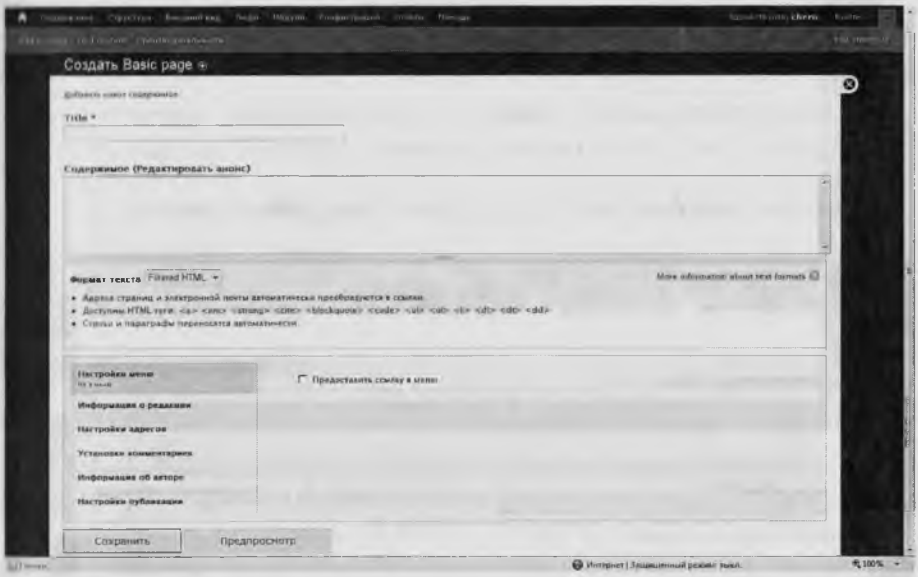
Перейдем в меню **К => Система => Global Redirect**. В принципе имеющиеся там настройки по умолчанию верны, и трогать ничего не надо. Просто посмотрите, что можно настроить.

Теперь зайдем в меню **Add content**. Пришло время посмотреть на то, как создаются материалы.

Нам предлагаются два типа материалов:

- **Article** (статья) — материал, который зависит от времени, например новость, пресс-релиз или запись в блоге;
- **Basic page** (Основная страница) — материал для формирования страницы со статическим (относительно) содержимым.

Создавать мы пока ничего не будем, поскольку нам нужно будет переносить информацию со старого сайта. Просто попробуем щелкнуть по варианту **Basic page**.



Размер поля **Содержимое** специально слегка поджали, чтобы все вместилось на экран. Итак, мы видим:

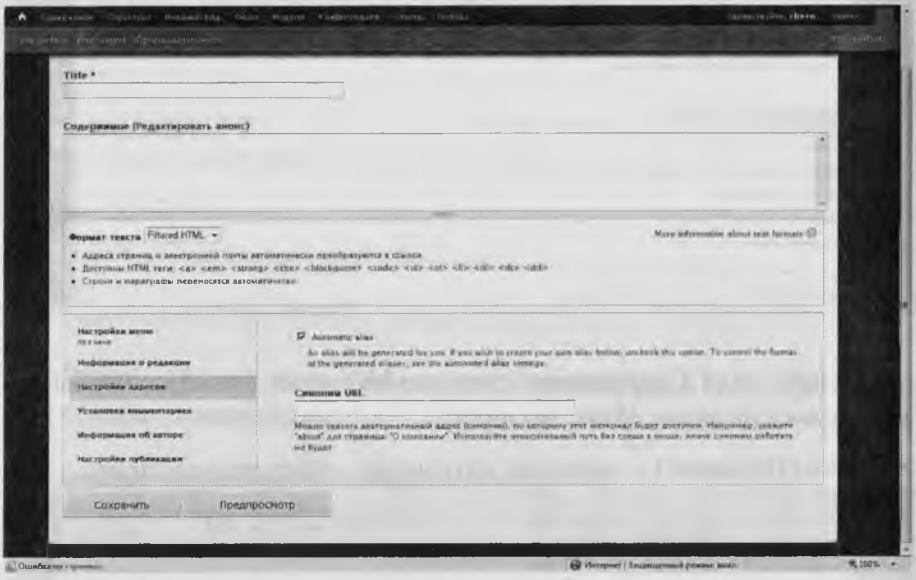
- **Title** (Название) — название материала — обязательное поле;
- **Содержимое** — собственно текст материала в HTML-формате (про анонс мы поговорим позднее);
- **Формат текста** — фильтр, который будет применен для содержимого данного материала.

Это базовые поля материала со смысловой информацией. Далее идет набор различных опций, сгруппированных по темам:

- **Настройки меню** — мы можем прямо отсюда поместить страницу с материалом в определенное месте меню (рассмотрим позже);
- **Информация о редакции** — служит для поддержки нескольких версий материала;
- **Настройки адресов** — этот пункт мы разберем ниже в данной главе;
- **Установки комментариев** — возможность оставлять комментарии к материалу;

- **Автор** — информация о публикации (имя автора и дата-время);
- **Настройки публикации** — тут три флажка:
  - **Опубликовано** — материал будет показан на сайте,
  - **Помещать на главную** — служит для формирования различных списков с анонсами, которые выносятся на главную страницу,
  - **Закреплять вверху списков** — материал особо интересен, и его надо поместить в списке одинаковых материалов наверху.

Давайте подробнее рассмотрим пункт **Настройки адресов**.



Поле **Синоним URL** дает нам модуль **Path**. В это поле мы можем ввести альтернативный адрес страницы. Как мы говорили, по умолчанию в Drupal все адреса страниц с материалами имеют вид node/id_node. А при переносе мы хотим использовать старые ссылки на страницы, поскольку, скажем, поисковые серверы помнят нас под этими адресами. Давайте введем в это поле, например, **main.htm**.

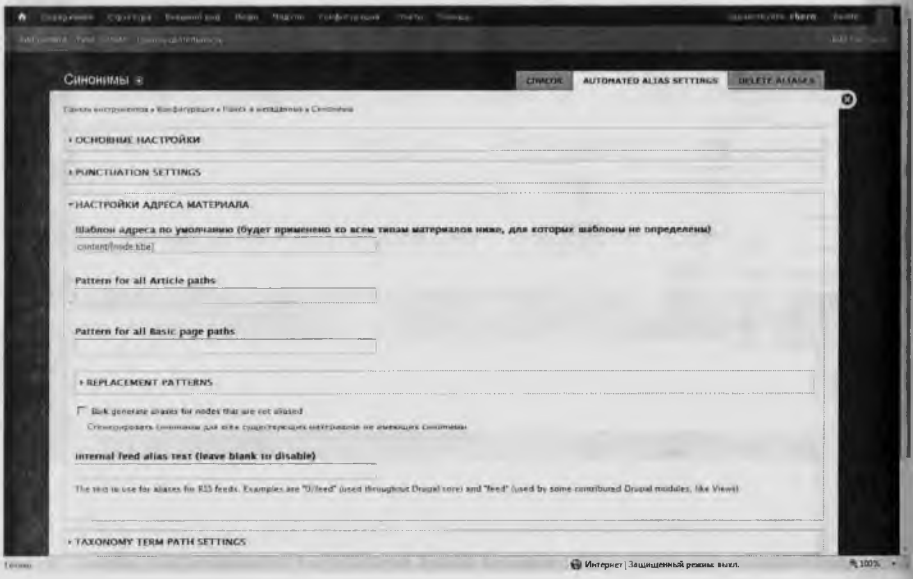
Поле недоступно! Почему?

Оказывается, его запрещает модуль **PathAuto**, который мы только что установили. Этот модуль позволяет автоматически формировать ссылки приличного вида. Поэтому, чтобы ввести нашу ссылку, нужно убрать чекбокс **Automatic alias** (Автоматические синонимы).

Посмотреть правила формирования автоматических синонимов мы можем, указав на линк **automated alias settings** (установки автоматических синонимов).

Опций тут огромное количество. Пока можете посмотреть, что еще можно поменять. В общем и целом все изначально настроено правиль-

по. В дальнейшем мы будем работать в основном с экраном **Настройка адреса материала**.



Здесь можно описать, каким образом формируется название для каждого типа материала, используя при этом все разнообразие подстановок из области **Replacement patterns**. Мы можем формировать ссылки на основе идентификаторов материалов или названий. Во втором случае ссылки будут более удобочитаемы. Модуль **Transliteration** при этом позволяет формировать английские ссылки на основе русских названий.

В результате при вводе материалов наши ссылки будут чистыми и понятными. Именно это мы и называем «красота»!

СОЗДАНИЕ НОВЫХ ТИПОВ МАТЕРИАЛОВ

- Добавление типа
- Настройка опций
- Дополнительные поля
- Настройка автоматических синонимов
- Разрешения для типа

Название материала — конечно, интересный перевод для слова «content», но оно мне постоянно напоминает 3D-студию. А теперь еще появляется и конструктор материалов.

Так и хочется использовать его, чтобы взять и «забабахать» ржавое железо или яркое серебро. Но наш конструктор просто позволяет создавать материалы новой структуры.

В Drupal 7 конструктор материалов (ранее — CCK, Content Construction Kit) встроен в ядро. Причем он встроен до такой степени, что его даже нельзя отключить.

Также расширились его возможности (скажем, поля можно добавлять не только материалам, но и пользователям).

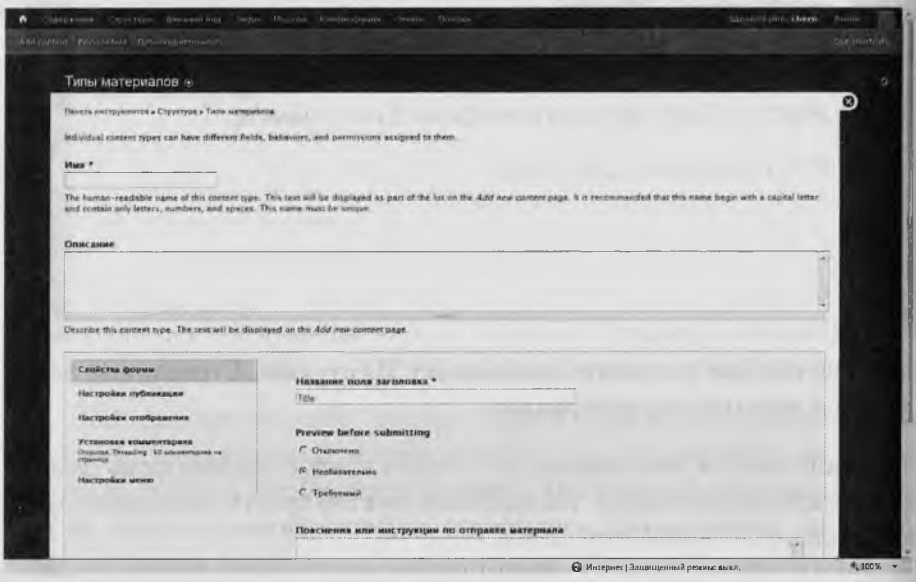
Поэтому, вместо того чтобы ставить уйму модулей и разбираться в их связях, мы можем сразу начать создавать новые типы материалов. Чем сейчас и займемся.

На нашем сайте есть отзывы. Мы сделаем их отдельным типом, а заодно потренируемся использовать **PathAuto**. Тип **Отзыв** весьма интересен тем, что он как бы виртуальный. Имеется в виду то, что достаточно часто одному материалу будет соответствовать одна страница на сайте.

Допустим, материал **Basic page** предназначен, чтобы сформировать одну страницу на сайте. В этом смысле материал **Отзыв** не предполагает просмотра в виде одной страницы.

Мы всегда будем показывать отзывы в виде страницы со списком. Так было на прошлом сайте.

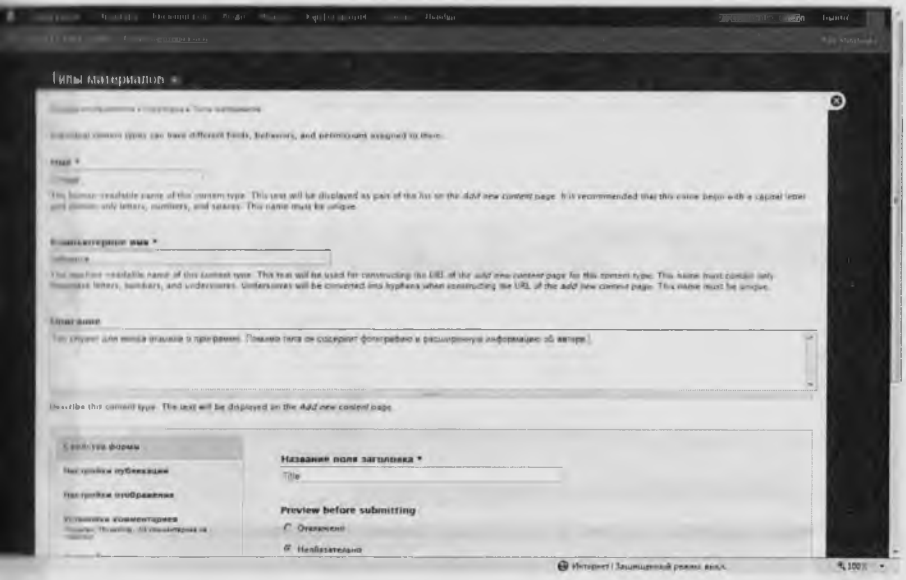
Для создания типа выполняем команду **Структура** ⇒ **Типы материалов**. Указываем вариант **Добавить тип содержимого**.



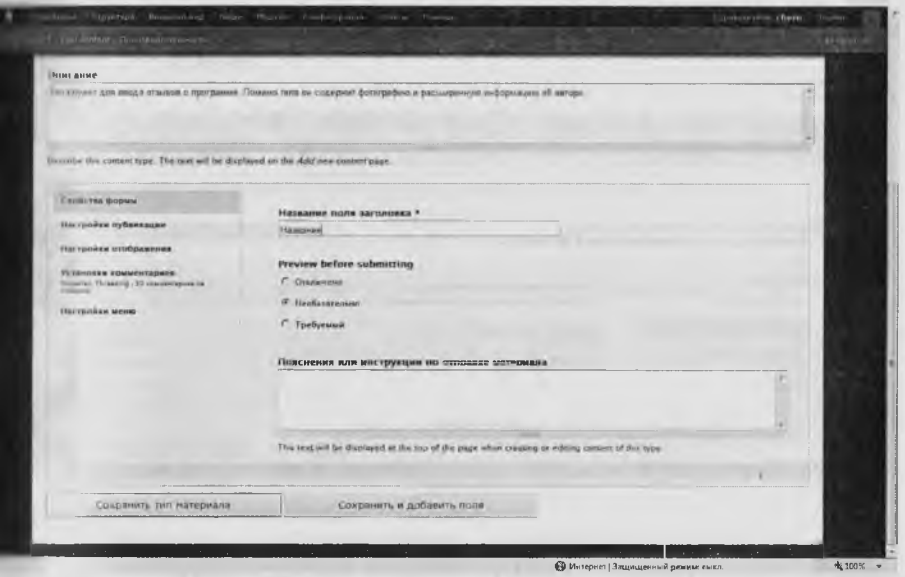
Имя по-русски — Отзыв. По-английски — reference. Нам нужно ввести оба варианта. Хитрый Drupal 7 ведет себя очень странно. Он показывает только поле **Имя**, а когда мы начинаем вводить в него информацию, справа отображается строка с надписью **Компьютерное имя** и кнопкой **Редактировать**. При этом кириллицу программа игнорирует и строку справа не показывает (по крайней мере, в этой версии).

Делаем просто. Вводим любую английскую букву и щелкаем по кнопке **Редактировать для компьютерного имени**. Потом вводим оба имени и описание. Оно ни на что не влияет — пишите, что хотите. Например:

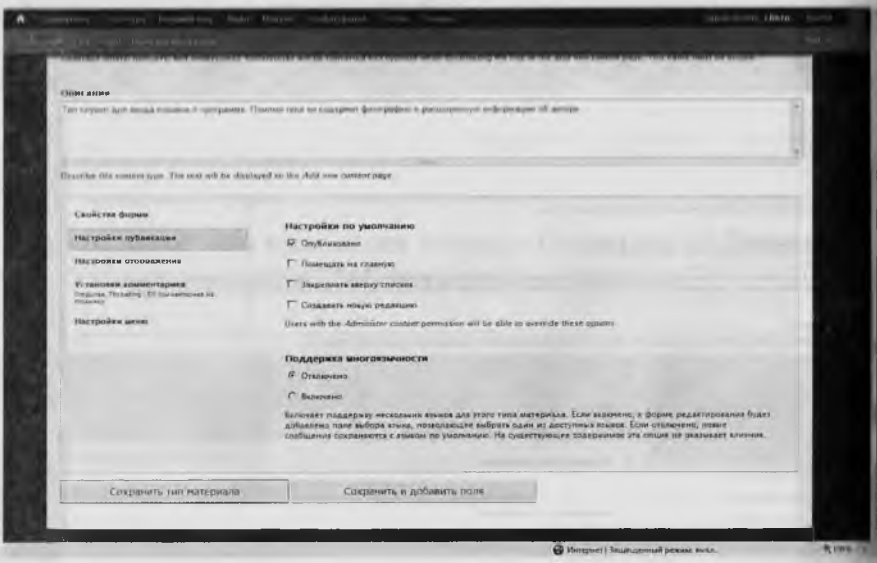
«*Тип служит для ввода отзывов о программе. Помимо тела он содержит фотографию и расширенную информацию об авторе*».



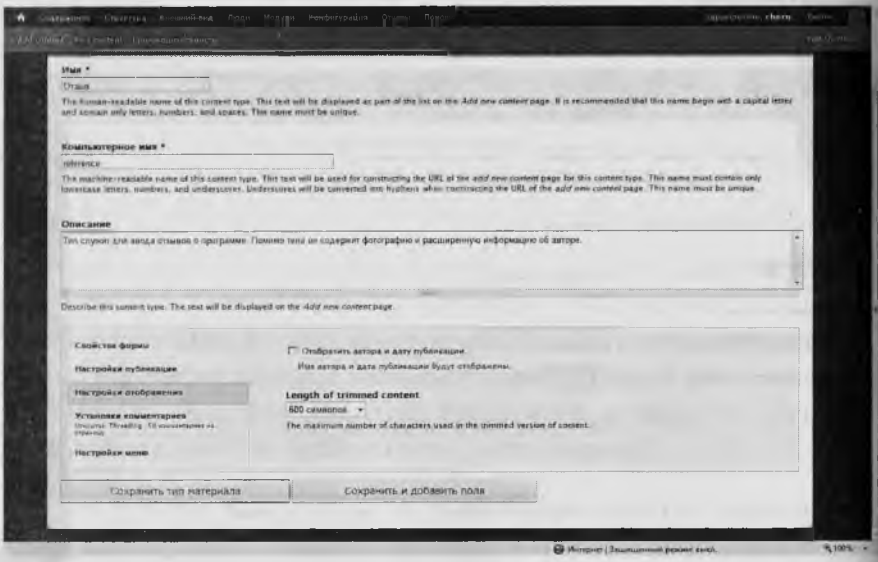
Далее следует множество групп опций. В свойствах формы мы ничего менять не будем. А нет — давайте переведем **Title** на русский. Напишем **Название**.



В настройках публикации уберем флажок **Помещать на главную**.

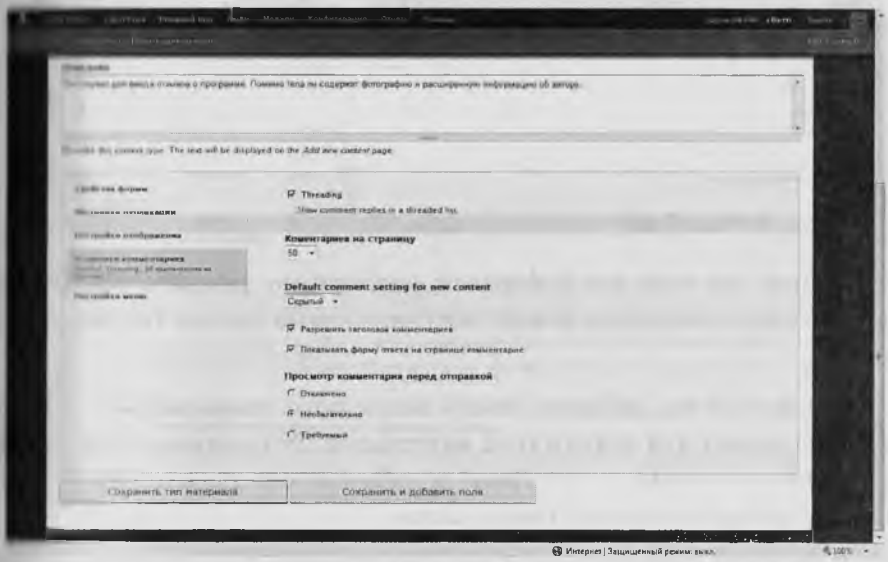


В настройках отображения отключим опцию отображения автора. Отзывы вводятся нами, а автора мы поместим в поле, поэтому отображать его не нужно.

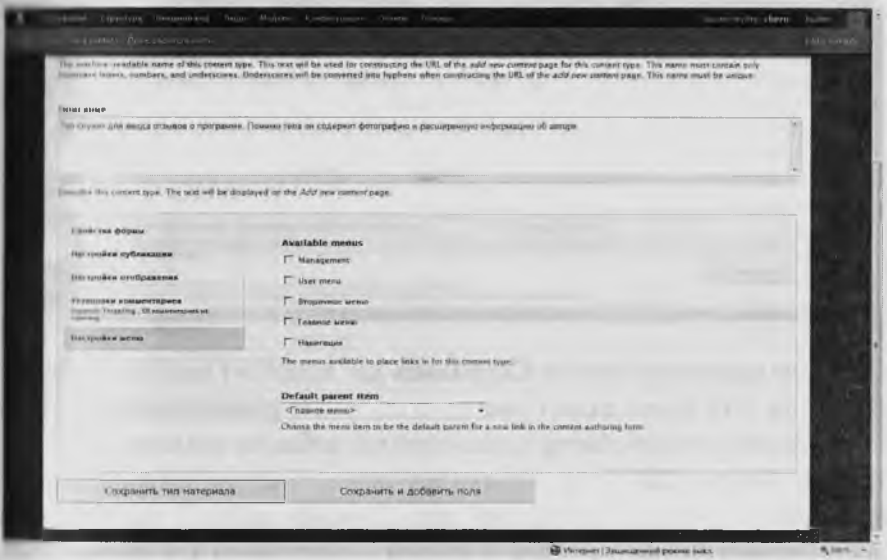


С установками комментариев все странно. Очень непонятный перевод для пункта **Default comment settings for new content** (Установки комментариев по умолчанию для нового материала). Опытным путем установлено, что если выбрать вариант **Скрытый** или **Закрыто**, то по

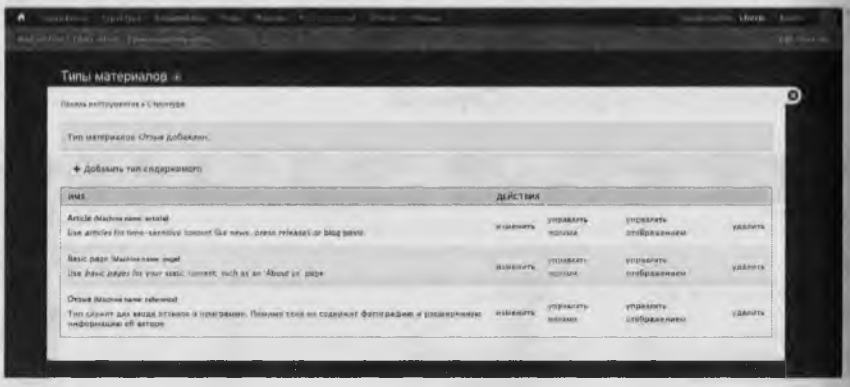
умолчанию комментарии будут запрещены. Я думаю, что со временем Скрытый начнет обозначать, что мы вообще не можем вводить возможность задания комментариев, а **Закрыто** — шанс выбрать эту возможность для конкретного материала (по умолчанию стоит вариант **Закрыто**). В общем, ставим флажок **Скрытый**.



Поскольку данный тип не выражается страницей, мы и не будем выделять его в меню.



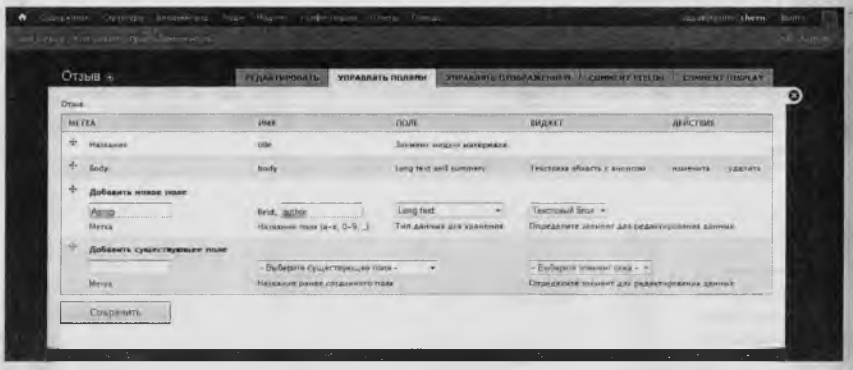
Ну все. Опции кончились. Сохраняем изменения.



Видим, что наш тип добавился; необходимо добавить обещанные поля. В списке напротив нового типа щелкаем по кнопке **Управлять полями**.

Мы можем как добавить совсем новое поле, так и выбрать поле из ранее заданных для любого типа материалов. Это довольно удобно при наличии однотипных полей. Давайте добавим поле для автора и используем существующее поле **Изображение**.

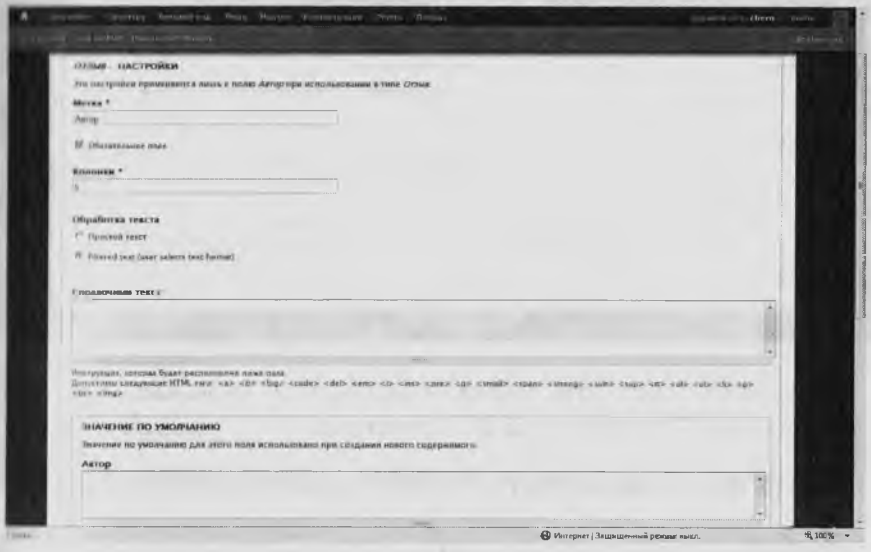
Вводим поле **author** типа **Long text** (Длинный текст). Элемент редактирования данных всего один — **Текстовый блок**.



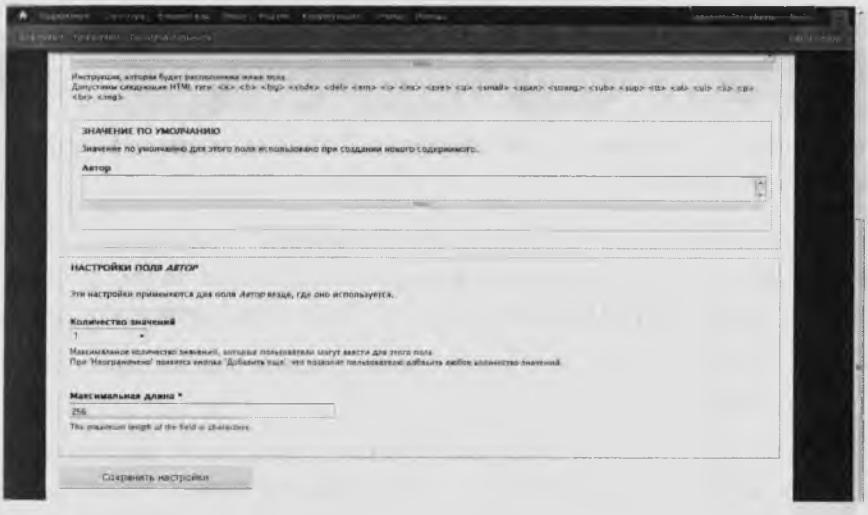
После щелчка по кнопке **Сохранить** нас попросят ввести длину данного поля. Эта длина задаст размер поля в базе данных и будет применяться ко всем полям **Автор** (даже если мы добавим это поле для другого типа). Введем **256**. Далее откроется экран со всеми настройками для поля **Автор**. Они разбиты на два типа. Первый набор — это настройки для поля внутри данного типа, а второй — настройки для данного типа

везде, где оно используется. Одна из настроек второго типа — это уже известная нам длина поля.

В настройках первого типа укажем, что мы используем **Filtered text** (Фильтрованный текст), и что поле обязательно.

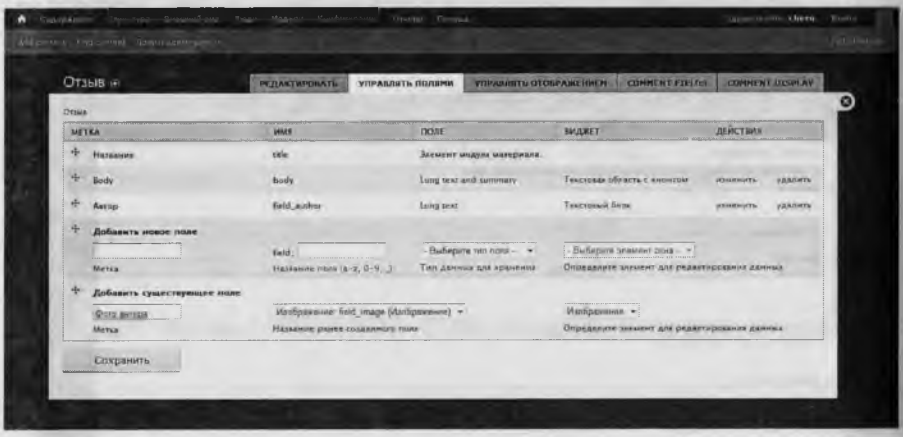


Настройки второго типа менять не будем.



Настройка каждого типа данных имеет свои особенности. Наиболее специфичны типы для картинок и файлов. Но в целом там все достаточно понятно.

Добавим еще **Фото автора**, используя существующий тип поля **Изображение**.



В опциях укажем, что это тоже обязательное поле. Остальные элементы оставим без изменений. Наш тип готов.

Теперь попробуем ввести несколько отзывов и посмотреть, что будет.

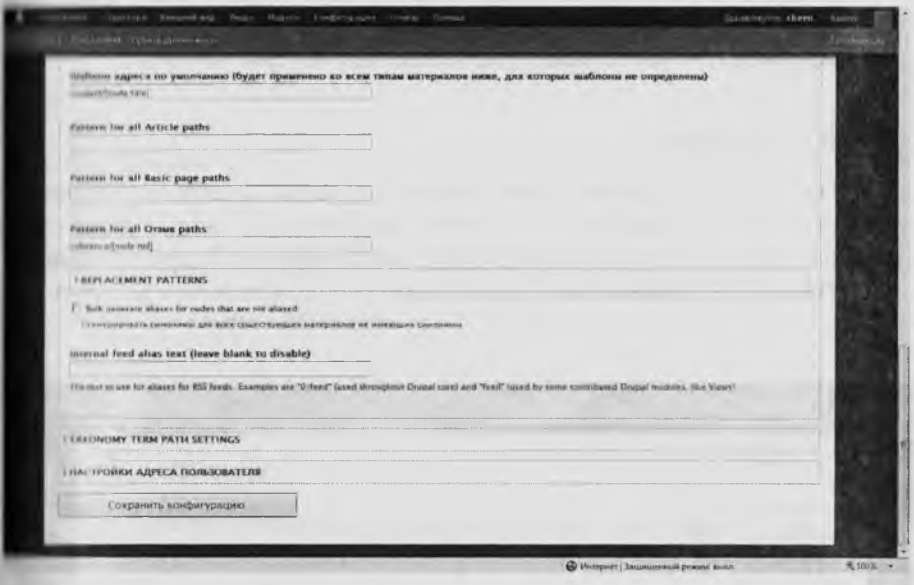
Вообще, неплохо было бы создать тестовый тип материалов и включить в него все возможные варианты полей (просто для пробы). При появлении новых типов полей мы их тоже будем добавлять.

Но до ввода отзывов нам надо настроить автоматические синонимы. Работа с ними ведется в меню **К ⇒ Синонимы**. Мы можем посмотреть список синонимов и настроить параметры их автоматического создания. Давайте перейдем в меню **Automated alias settings ⇒ Настройки адреса материала**.

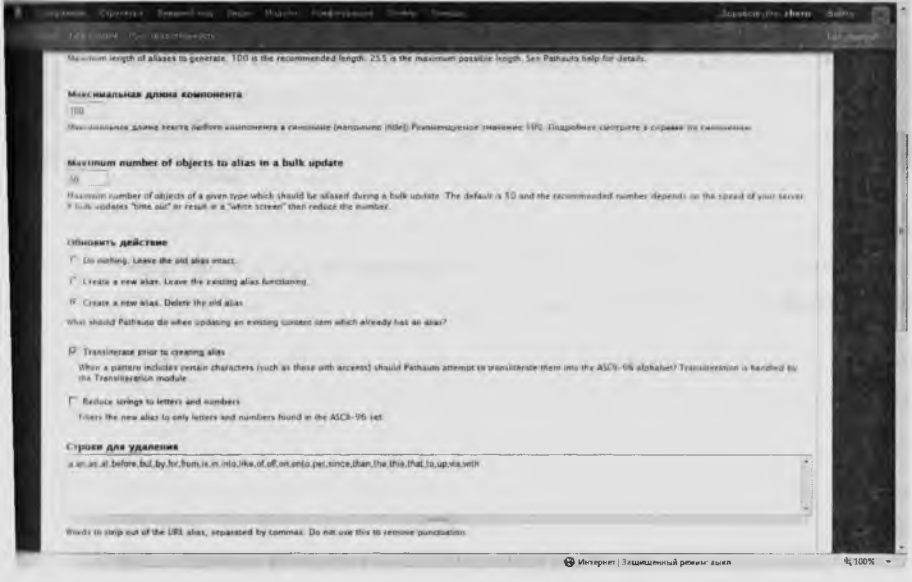
Для каждого типа материала можно ввести свой шаблон. Также можно ввести общий шаблон по умолчанию. Шаблон по умолчанию — **content/[node:title]**. Это влечет за собой определенные последствия:

- надо включить транслитерацию, чтобы русские имена переводились на английский;
- есть вариант включить автоматические синонимы в виде **content/[node:nid]**, в этом случае транслитерация не нужна;
- для шаблонов **Картинки** и **Отзывы** следует использовать синонимы с **[node:nid]**, поскольку они не будут выведены в виде страницы и не важно, какой у них URL.

Установим для отзывов **reference/[node:nid]**. А по умолчанию пусть будет **content/[node:title]**.



Установим транслитерацию (**Transliterate prior to creating alias** тут нет, но в группе **Основные настройки**).

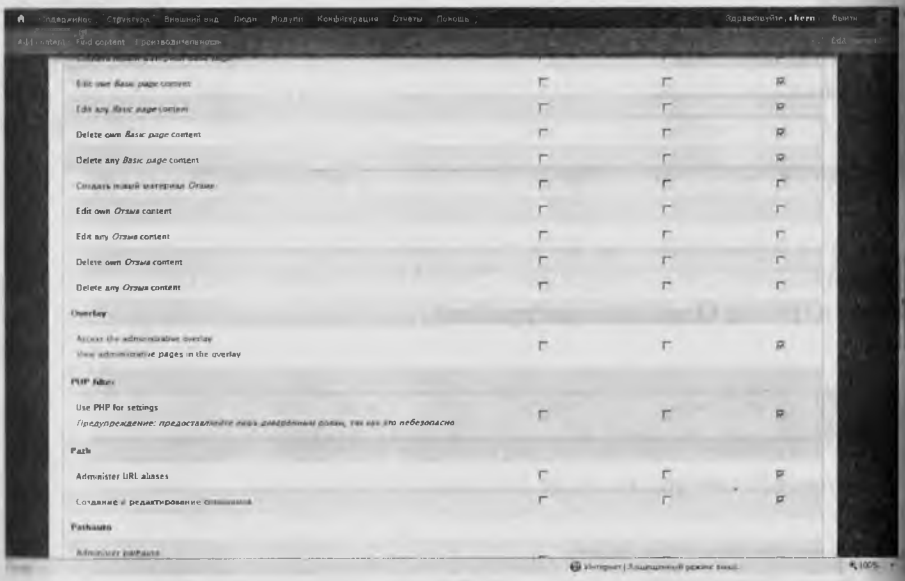


ПРИМЕЧАНИЕ

Введя часть материалов, можно удалить синонимы для них и пересоздать синонимы автоматически, указав опцию **Bulk generate aliases for nodes that are not aliased** (Массовое формирование синонимов для материалов, у которых еще нет синонимов).

ВНИМАНИЕ

При создании нового типа материалов в разрешениях появляются строки с правами на этот тип. При этом роли administrator права автоматически не выдаются. Необходимо выполнить команду **Люди** ⇒ **Разрешения** и самостоятельно дать права на этот тип.



Отмечаем все пустые чекбоксы в правой колонке и щелкаем по кнопке **Сохранить права доступа**.

СОЗДАНИЕ МАТЕРИАЛОВ

- WYSIWYG
- IMCE
- CKEditor и их настройка
- Хранение картинок

Создавая материалы, мы можем задавать содержимое полей в виде HTML. Это относительно просто, но не всем удобно. Все уже привыкли работать в WYSIWYG-редакторах (What You See Is What You Get — что вижу, то и имею).

Поэтому для Drupal предлагается набор различных редакторов, которые позволяют форматировать текст и потом выдавать его в формате HTML.

Эволюция редакторов для Drupal была довольно длительной. Раньше все использовали FCKEditor (http://drupal.org/project/fckeditor), а потом вышла его новая улучшенная версия — CKEditor (http://drupal.org/project/ckeditor).

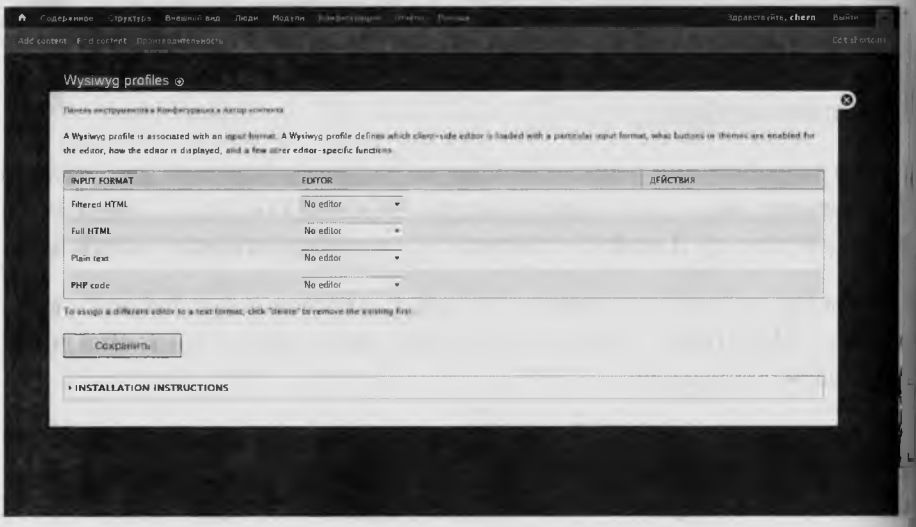
Но разработчики решили вообще не создавать модулей для отдельных редакторов — так появился модуль WYSIWYG (http://drupal.org/project/wysiwyg). Давайте установим его и разберемся, как он работает.

Мы также сразу активируем модуль для организации файлов и картинок — IMCE (http://drupal.org/project/imce). Итак, скачиваем и устанавливаем оба модуля.

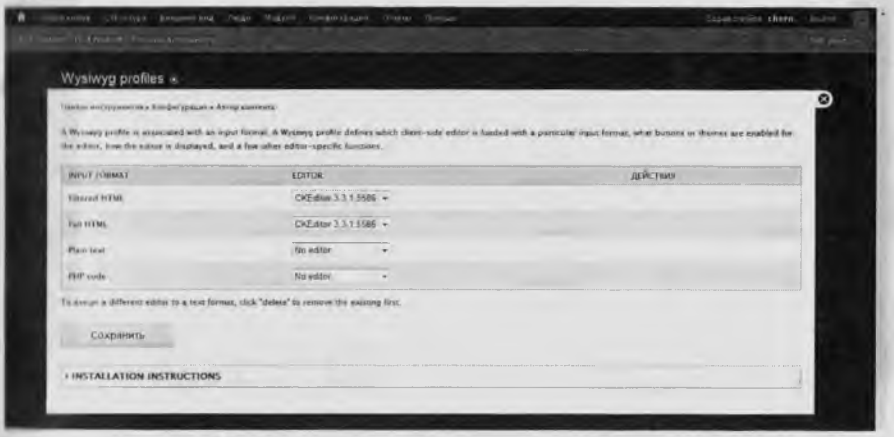
Заметим, что для CKEditor нужно скачать не только модуль, но и сам редактор. Алгоритм действий описан на странице настройки этого модуля: **К ⇒ Автор контента ⇒ Wysiwyg profiles**.

Мы установим CKEditor. Для этого нужно скачать архив и распаковать его в директорию **sites/all/libraries/ckeditor**. Лучше скачивать в формате tar.gz, поскольку почему-то через ftp не проходит файл .htaccess. Распаковывается архив командой **tar -xf ckeditor_3.3.1.tar.gz**. Может у вас уже будет более новая версия.

Теперь, переместившись в **К ⇒ Автор контента ⇒ Wysiwyg profiles**, мы должны увидеть следующее:



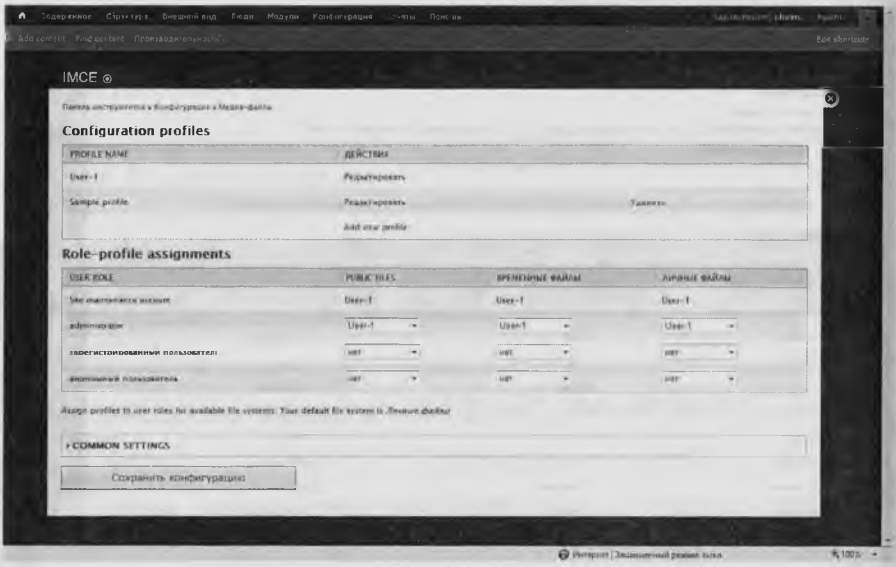
Можно задать различные редакторы для разных форматов ввода. В самом деле, зачем нам HTML-редактор, если мы должны ввести **Plain text** (Простой текст)? Поэтому выберем редактор только для ввода HTML-текста.



После сохранения появится возможность настраивать опции для форматов ввода, где установлен редактор.

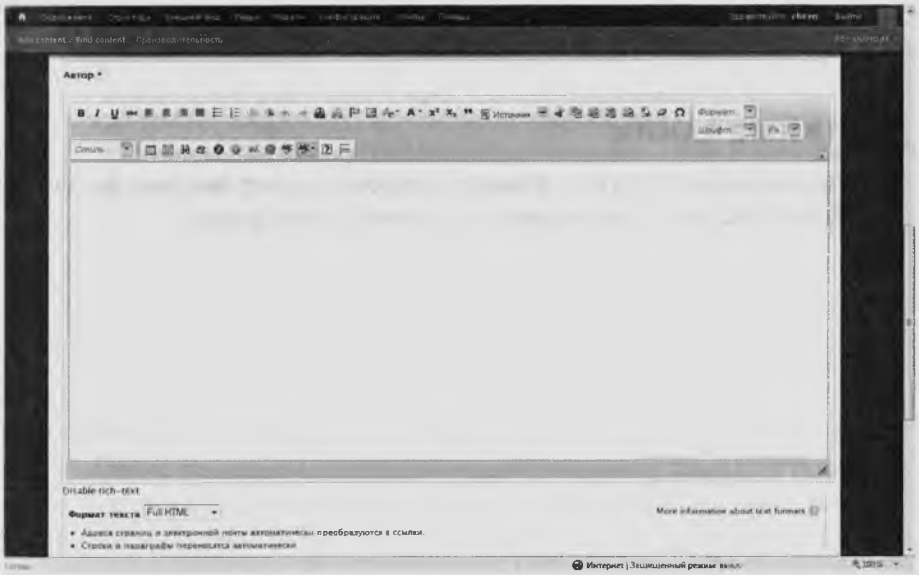
Необходимо установить язык **ru_UTF-8** и поставить все флажки в разделе **BUTTONS AND PLUGINS** (если их не включить, редактор не появится). Для **Filtered HTML** мы также должны установить язык, но активировать только некоторые опции.

Также необходимо подключить IMCE: выполняем команду **К ⇒ Медиа файлы ⇒ IMCE** и устанавливаем для роли **администратор** профиль **User-1**.



Наконец-то мы сможем создать свою первую страницу. Давайте сформируем три отзыва и одну страницу и посмотрим, как их опубликовать в нужном формате. Для создания отзывов переходим в **Add content ⇒ Отзыв**.

В полях **содержимое** и **Автор** появился наш редактор в режиме Filtered HTML. Если переключить режим на Full HTML, отобразится полный CKEditor с максимальными возможностями.



Вместо обычного текстового поля теперь везде будет CKEditor. Иногда это хорошо, иногда плохо. Обычно для полей это плохо, поэтому мы используем возможность отключить CKEditor для определенных по-

лей. Под обоими полями с CKEditor отображаются специальные надписи с указанием идентификатора для включения и отключения данного элемента (**Disable rich-text**). К сожалению, сейчас это нельзя применить для конкретного случая, но мы надеемся, что модуль WYSIWYG будет совершенствоваться, и мы увидим эту возможность как и в модуле для CKEditor.

Итак, переходим к созданию материалов и создаем для пробы несколько образцов. Формируем отзывы (вставляем имя автора, картинку и т. д.) и страницы. Страницы ОБЯЗАТЕЛЬНО надо создавать на основе страниц, существовавших ранее. Нужно создавать страницы с уже имеющимися URL, иначе это сильно повредит посещаемости сайта. Вы можете создавать синонимы вида main.php или references.htm — все они будут работать в Drupal.

Желая поместить картинку в **Отзыв**, мы просто загружаем ее с диска в соответствующее поле. Помимо этого мы можем хранить картинки различными способами:

- картинки оформления — относятся к теме и хранятся вместе с ней;
- картинки поля — помещаются так же, как и для отзыва;
- картинки внутри материалов — импортируются при помощи IMCE и размещаются по пути /system/files.

ПРИМЕЧАНИЕ

При вставке HTML не забывайте включать пункт **Формат ввода** = **Full HTML**, иначе вы не увидите разметку при показе.

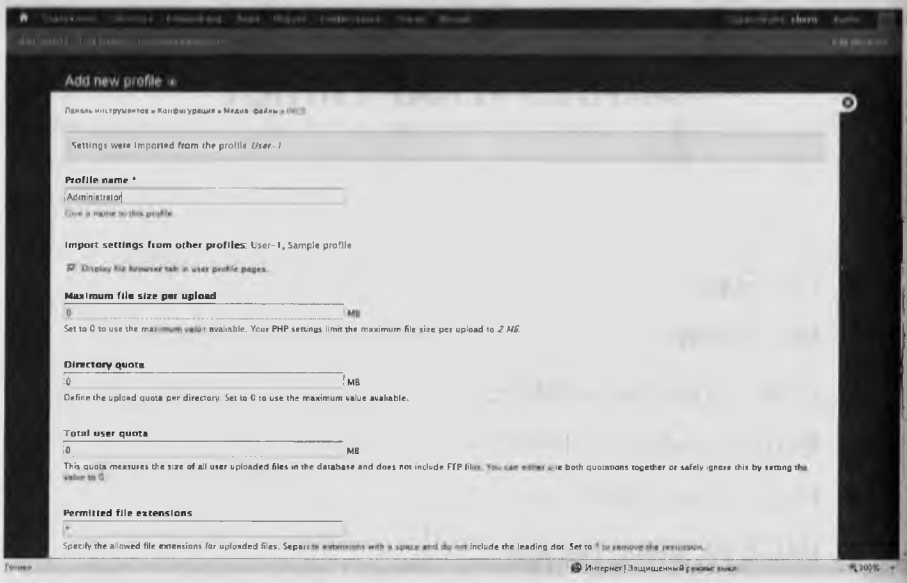
ФАЙЛЫ И КАРТИНКИ

- File Force
- MCE Mkdir
- Новый профиль IMCE
- Вход и работа с IMCE
- Настройка File Force
- IMCE Wysiwyg bridge
- Вставка картинки предпросмотра
- Открытие основной картинки в модальном окне

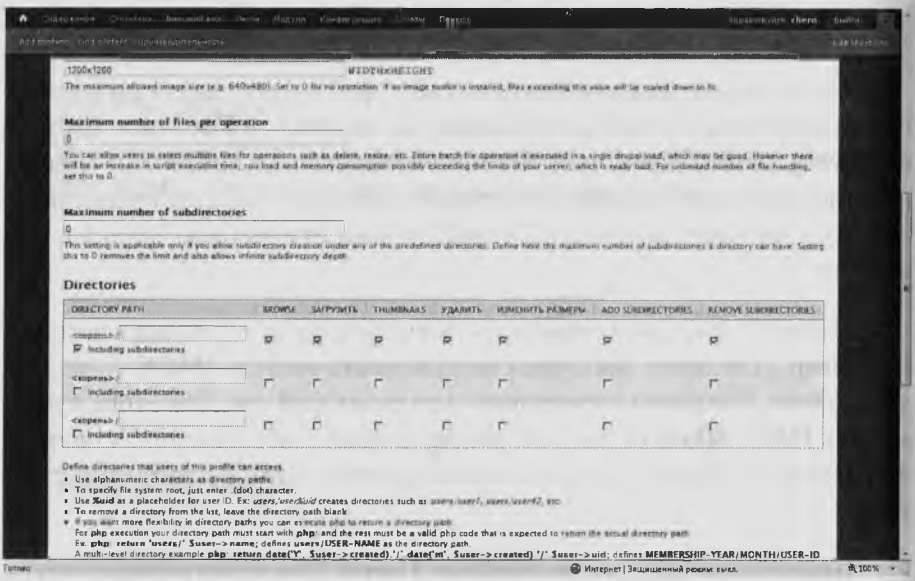
Иногда мы хотим разместить на сайте файлы для скачивания всеми пользователями. Самый простой модуль для этого — **File Force** (http://drupal.org/project/file_force). На момент написания книги этот модуль существовал только в версии для Drupal 6. Поскольку модуль очень прост, надеемся, он будет готов для Drupal 7 к моменту выхода официальной версии.

Однако, прежде чем позволить скачивать файл, его нужно закачать на сервер. Для этого мы будем использовать модуль **IMCE** — он уже установлен. Создадим директорию для своих файлов. Для этого нужен модуль **IMCE Mkdir** (http://drupal.org/project/imce_mkdir). Установив и включив его, надо создать отдельный профиль IMCE для администратора.

Импортируем данные из профиля **User-1**. После установки модуля **IMCE Mkdir** появятся две новые возможности: **Add subdirectories** (Добавить поддиректорию) и **Remove subdirectories** (Удалить поддиректорию). Отметим их.



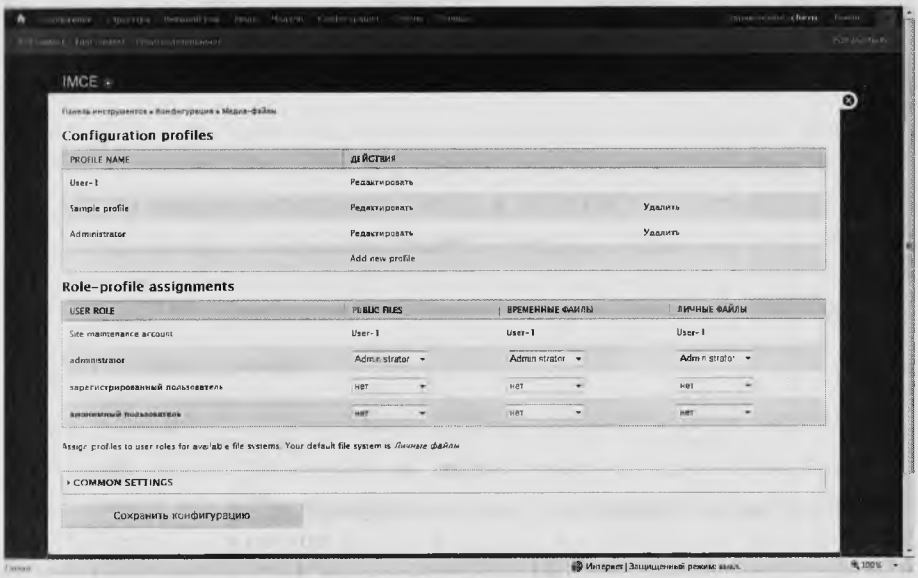
Также неплохо убрать ограничение на число поддиректорий — ставим **0** в поле **Maximum number of subdirectories**. И, конечно, нужно дать права системной роли adminisrator на **IMCE**.



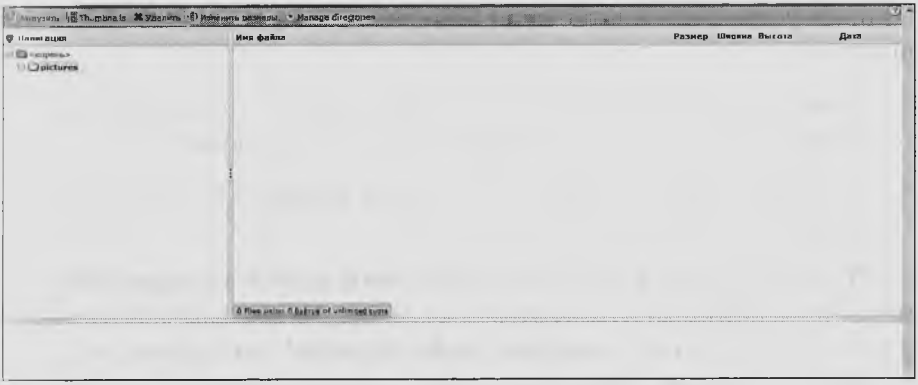
В принципе можно оставить и профиль **User-1**. Тогда вы сразу получите возможность работать с поддиректориями.

Итак, мы все настроили и можем заходить в **IMCE**. Для этого можно просто набрать http://www.имя.сайта/imce.

Получим такой вид.

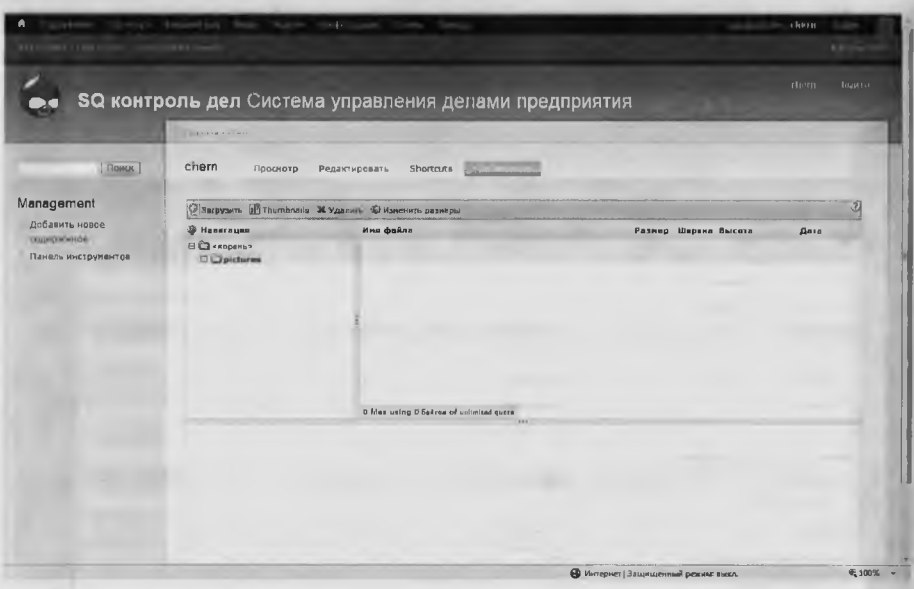


Можно зайти и со своей страницы. Для этого закроем панель администрирования и укажем свое имя справа вверху. В меню выберем **File browser** (Менеджер файлов).



Лучше просто открыть менеджер IMCE в отдельной закладке.

Теперь создадим директорию **sqrus** и разместим в ней несколько файлов.



Надо зайти в директорию **sqrus**, щелкнуть по кнопке **Загрузить** и поместить туда все файлы, которые предназначены для прямого скачивания. Ссылки на данные файлы в нашем примере — /sites/default/files/sqrus/имя файла. Но при попытке скачать файл по этой ссылке мы обнаружим, что csv-файл предлагает сохранить информацию на диск, а xml-файлы пытаются открыться напрямую в браузере. Чтобы этого избежать, нам и нужен модуль **File Force**.

Чтобы задействовать данный модуль, нужно (сначала, конечно, его надо установить и включить) поменять ссылку на один из двух вариантов:

- */download/sites/default/files/sqrus/имя файла* — это официальный вариант использования, который предложен в документации;
- */download/sqrus/имя файла* — это тоже работает, но выдает предупреждение.

После этого все включится само (то есть модуль автоматически заменит путь download нужным), и браузер предложит вам либо открыть файл (в некотором приложении), либо сохранить его на диск. Это, собственно, нам и нужно. Что касается двух вариантов вызова, то создатель модуля декларировал, что с версии 6.x-1.1-beta3 нужно использовать первый вариант. Тем не менее второй вариант тоже работает, и мне он нравится больше, поскольку ссылки получаются короткими и не дают хакерам данных о структуре наших директорий. А чтобы предупрежде-

ние не появлялось, достаточно закомментировать строку, в которой оно возникает. В версии модуля 6.x-1.1 это строка 46, которая определяет размер файла. Надо сказать, что и без нее все работает, и Drupal как-то сам определяет размер файла и выдает его.

После этого исправления нельзя будет обновлять модуль через FTP путем скачивания, но можно будет обновлять через CVS, конечно, каждый раз проверяя на наличие конфликта.

Еще один интересный момент возникает при попытке вставить в текст картинку. Это делается просто при помощи CKEditor. Однако необходим модуль интеграции IMCE и WYSIWYG — это IMCE Wysiwyg bridge (http://drupal.org/project/imce_wysiwyg). После установки он появится в настройках **wysiwyg** редактора как один из BUTTONS AND PLUGINS. Необходимо включить его для режима Full HTML.

Рассмотрим пример.

Итак, выбираем место, в которое хотим вставить картинку, и щелкаем по кнопке вставки. Открывается окно для вставки картинки. Первая закладка называется **Информация об изображении**. В нее мы вставляем URL-изображения. Это, как правило, уменьшенная копия основного изображения. Она может быть сформирована автоматически при импорте изображения (нужно просто указать размеры копий, которые мы хотим выбрать) или создана вручную. Щелчок по кнопке **Просмотреть на сервере** позволяет выбрать картинку при помощи **IMCE**.

Чтобы описать основное изображение, используется вторая закладка (**Ссылка**).

Здесь мы вводим URL основной картинки и указываем, каким образом будем открывать полное изображение.

В результате порождается простейший код, например:

```
<a href="/system/files/large.gif" target="_
blank"><img alt="" src="/system/files/small.gif"
style="width: Wpx; height: Hpx" /></a>
```

В общем, больше ничего и не нужно. Можно и самостоятельно вставить подобный код в источник:

```
<img onclick="window.showModalDialog('/sys-
tem/files/large.gif','', 'dialogWidth=LWpx;
dialogHeight=LHpx');"

 src="/system/files/small.gif" style="width: Wpx;
height: Hpx; cursor: pointer" />
```

Тогда картинка отобразится в модальном окне.

МОДУЛЬ VIEWS

- Views
- Ctools
- Advanced Help
- Помощь по Views
- Создание view
- Описание полей
- Предпросмотр
- Фильтры
- Новый вид показа — **Страница**
- Просмотр получившейся страницы

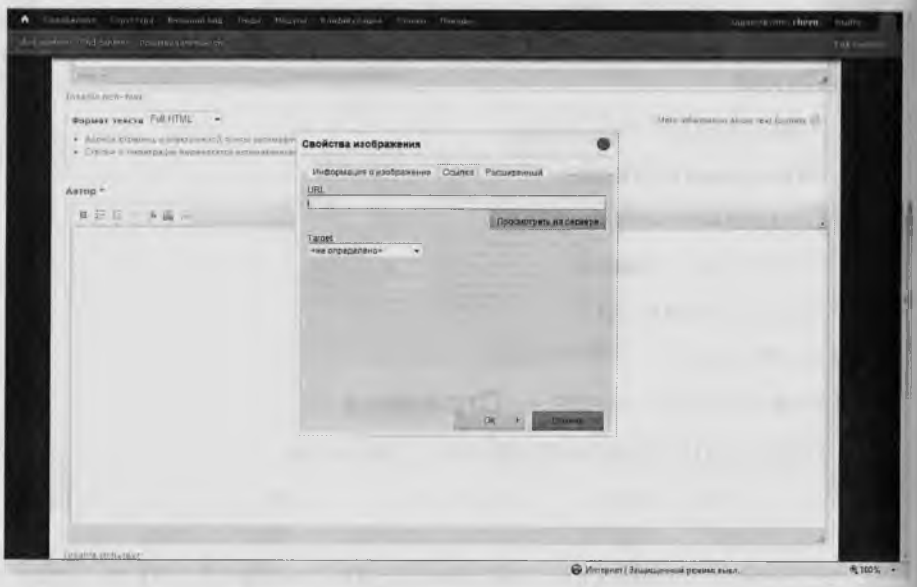
Создав часть материалов, нам надо понять, как они будут представлены на экране. Для этого необходимо настроить модули для показа:

- сам модуль **Views** (http://drupal.org/project/views) предназначен для создания выборок материалов и их отображения. По сути это SQL для материалов без использования программирования. Конечно, совсем без программирования мы не обойдемся, но выборки гораздо проще и эффективнее делать через **Views**;
- модуль **Ctools** (http://drupal.org/project/ctools) открывает широкий спектр разнообразных возможностей, но мы не знаем пригодятся ли они нам в данном случае;
- **Advanced Help** (http://drupal.org/project/advanced_help) улучшает помощь и требуется для модуля **Ctools Plugin Example**.

Итак, ставим все три модуля, включаем все разрешенные опции (кроме примеров — потом посмотрите их отдельно) и пробуем использовать разумно то, что у нас получилось.

Сначала сформируем список отзывов. Для этого нам нужен модуль **Views**. Работа с модулем проводится в меню **Структура** ⇒ **Views**, а помощь можно найти в пункте **Advanced Help** ⇒ **Views**.

Для начала надо прочитать **Getting Started**. Без **Advanced Help** нам было бы очень плохо, поскольку нормальный блок советов по **Views** не отображался бы. А так открываем **Getting Started** и изучаем картинку, чтобы понять, что к чему.

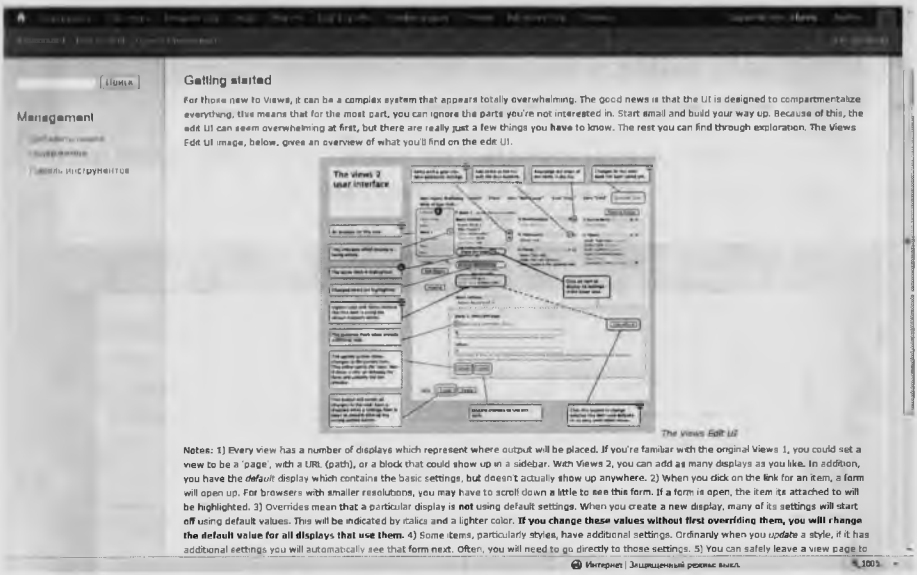


Идея проста — есть общие параметры запроса. Для их редактирования нужно выбрать один из параметров, после чего он появится в области внизу. Сохранение параметра производится соответствующей кнопкой в этой области. Общее сохранение происходит по кнопке **Save** (Сохранить) внизу.

Для одного **View** можно определить несколько видов показа. Каждый вид может отличаться набором полей и опциями. Допустим, мы хотим отобразить как общий список отзывов, так и список в блоке на странице. Это можно сделать при помощи одного **View**.

Создадим один **View** для показа отзывов. Переходим в меню **Структура ⇒ Views**, щелкаем по кнопке **Add new view** (Добавить) и вводим название и описание для него.

Можно добавить тег, чтобы разделить **View** по признаку сайта. **View** мы формируем на основе материала (используйте в названии и теге только латинские буквы и цифры!). Отмечаем его и щелкаем по кнопке **Next** (Далее).



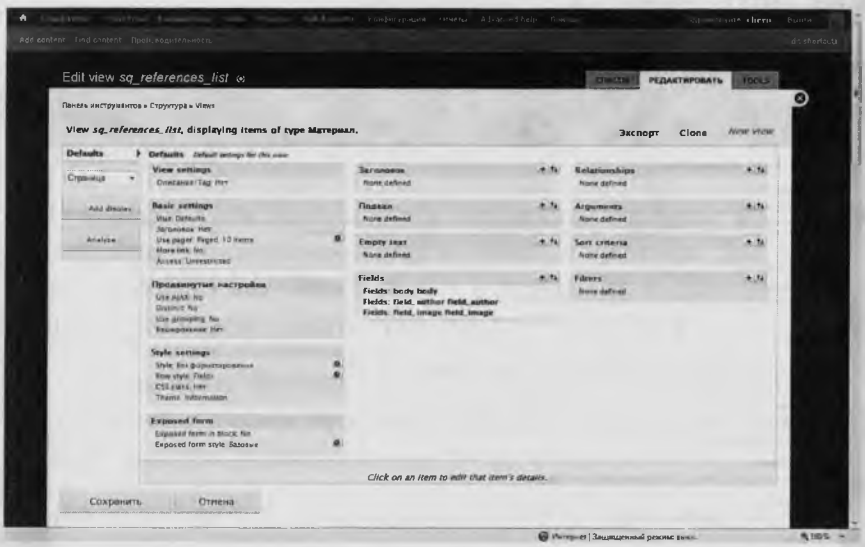
После этого мы попадаем на тот самый большой экран, вид которого изучали в директории **Помощь**.

Сначала необходимо определить выводимые поля. В заголовке группы **Fields** (Поля) щелкаем по кнопке со знаком плюс и выбираем группу **Fields**. Можно выбрать все три поля сразу:

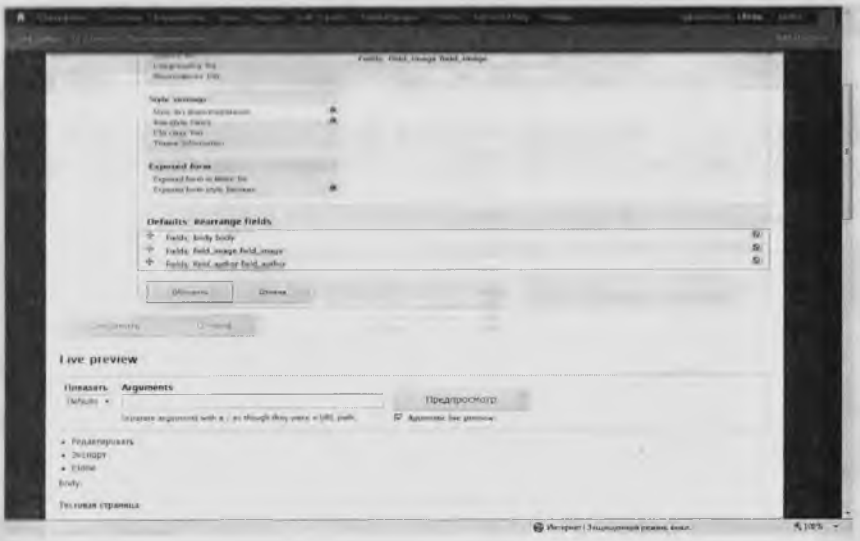
- **Fields: body** — текст отзыва;
- **Fields: field_author** — подпись под картинкой;
- **Fields: field_image** — картинка автора.

Щелкаем по кнопке **Добавить**. Нас начнут спрашивать про настройки для всех полей — оставляем все по умолчанию.

Итак, мы добавили поля, но их надо упорядочить. Щелкаем по кнопке со стрелочками в группе с полями.



Как обычно, все отобразилось в области внизу. Меняем, перетаскивая стрелочки в нужном порядке.



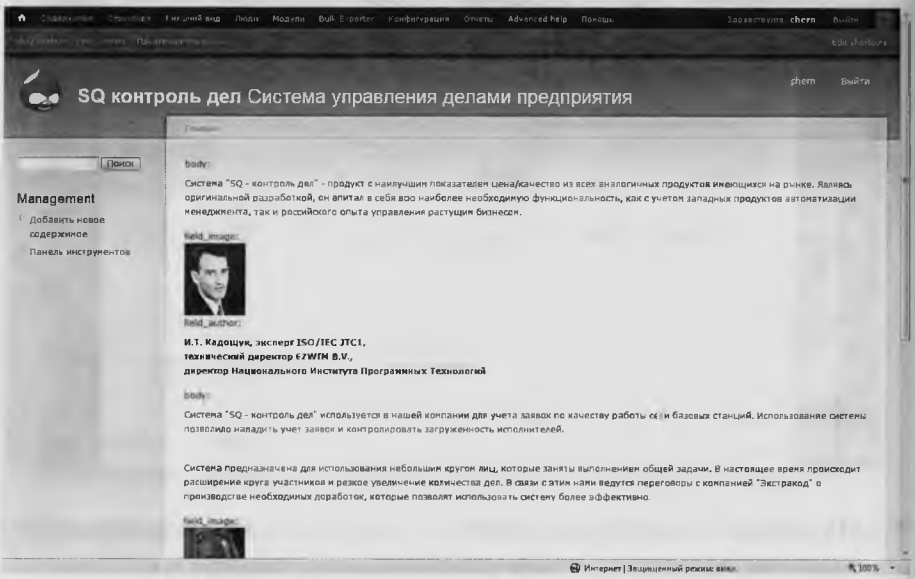
Щелкаем по кнопке **Обновить**. И смотрим, что получилось: внизу автоматически появилась страница с предпросмотром. На ней отобразились отзывы, а также все остальные страницы. Непорядок. Надо делать фильтры.

В **Фильтры** мы добавляем два фильтра:

- **Материал: Тип = Отзыв** — чтобы выбрать только отзывы;

- **Материал: Published or admin** — будем показывать только опубликованные материалы, чтобы управлять ими.

Однако пока это было лишь определение Defaults. Чтобы можно было увидеть что-то на сайте, нужно создать вид показа (display). Выбираем тип **Страница** и вводим **Add display**. Слева появляется закладка для нового режима просмотра — **Страница**. Нам нужно ввести URL, по которому будет показываться данная страница. Выбираем закладку **Страница** и **Page settings** — **Путь**. Вводим **references** и сохраняем **View**.



Теперь идем на страницу **references** — вводим ее после имени сайта — и проверяем, что на ней отображается список отзывов.

Итак, мы научились создавать сложные страницы из набора материалов путем создания **View**. Заметим, что в меню администрации мы перемещались от меню **Конфигурация** к меню **Содержимое**, а сейчас уже переместились к меню **Структура**.

ПРИМЕЧАНИЕ

Есть еще полезный модуль **Panels** (http://drupal.org/project/panels). Он позволяет разбивать как страницу, так и области страницы на подобласти. Мы можем без программирования создать множество различных видов разметки страниц . Говорят, правда, что лучше для разметки использовать темы. Пока запомним это и завяжем узелок на будущее. Мало ли, вдруг пригодится.

СОЗДАНИЕ ТЕМЫ

- Структура страницы
- Файлы темы
- тема.info
- Регионы
- Пример файла тема.info
- page.tpl.php
- Пример файла page.tpl.php
- Другие файлы темы
- Установка темы
- Проблемы с совместимостью стилей
- Пример вывода центральных областей

Итак, мы научились описывать и создавать материалы, а также делать из них выборки. Пора попробовать расположить все это на сайте.

Структура страницы сайта раньше была следующей:

- заголовок — содержит ссылку на сайт компании;
- логотип;
- блок контактов и навигации (было плоское выпадающее меню) — может заменяться блоком для показа скидки;
- тело (структура его зависит от страницы);
- нижняя часть (содержит три основные навигационные кнопки);
- счетчики.

Тело бывает разных видов и может содержать:

- просто текст страницы;
- подменю из нескольких пунктов для переключения между страницами;
- содержание центральной страницы — навороченный HTML с заставкой и меню специального вида.

Страницы единообразны и состоят из различных специальных кусочков, причем различия могут быть как на первом, так и на втором уровне. То есть структура страницы едина, но отличие возникает на втором уровне — в одном из блоков. Сразу думается, что может понадобиться модуль **Panels**. Ладно, будем пробовать...

Итак, наша задача — создать тему с указанными выше областями и нашим оформлением. Внутренность тела мы потом будем создавать при помощи панелей.

Самый простой способ изучить построение тем — это разобраться в устройстве существующих. Но отметим, что мы создаем тему для конкретного сайта, и нам не нужно проводить ее полную универсализацию.

Наконец, мы добрались до написания буковок, а не просто ввода данных через Интернет. Чтобы создать тему, нужно сформировать определенное количество файлов. В принципе есть редактор тем — **Theme Editor** (http://drupal.org/project/theme_editor), но он актуален, если у вас есть проблемы с помещением тем на сайт. Обычно проще держать тему у себя на диске или под CVS и копировать ее на сайт через FTP.

Давайте создадим простейшую тему и будем дополнять ее по мере необходимости. Для начала нужно сформировать набор из нескольких стандартных файлов в директории с именем темы:

- `/наша_тема`;
- `/images` — директория для картинок;
- `logo.png` — лого нашего сайта;
- `page.tpl.php` — php для показа страницы; его мы будем редактировать в первую очередь;
- `screenshot.png` — маленькая картинка нашей темы; сделаем уменьшенный скриншот старого сайта;

- `наша_тема.info` — файл описания темы в формате обычного ini-файла;
- `style.css` — стили нашего сайта.

Этого хватит, чтобы сделать первую тему. Выберем имя темы (латиница без пробелов) и создадим структуру. Подготовим все картинки и стили. Стили просто взяли со старого сайта. Остались два непонятных файла: page.tpl.php и наша_тема.info.

Основой темы является файл наша_тема.info. Он содержит основные данные о теме в формате ini-файла. В нем описывается набор стандартных параметров и два массива: `regions` и `features`. Нужны следующие параметры:

- `name = развернутое название темы`;
- `description = Развернутое описание темы`;
- `version = 1.0` — нужно только для публикации тем;
- `core = 7.x` — мы работаем на Drupal 7;
- `engine = phptemplate` — новички ставят только так;
- `stylesheets[all][] = style.css` — можно описать несколько файлов стилей;
- `scripts[] = mm_images.js` — можно описывать файлы скриптов.

Приведем и несколько ненужных параметров:

- `screenshot = другое имя скриншота`;
- `base theme = служит для наследования тем` — штука редкая и не особо нужная;
- `php = минимальная версия PHP` — обычно не нужно ставить; тоже важно только для публикации.

Записываем нужные параметры в наш файл. В принципе этого уже достаточно. Но, кроме этого, в .info-файле описывается набор визуальных элементов (features) и областей (regions). Элементы мы трогать не будем (менять их список полезно только для публикуемых тем), а области разберем подробнее.

Страница в Drupal состоит из областей и блоков. Блоки могут помещаться в области, а могут напрямую выводиться на странице. Видимостью блоков на странице можно управлять. Типичный способ создания

блоков — создание **View** типа «блок». Стандартные темы имеют возможность размещать блоки в следующих областях:

- Заголовок = Header;
- Левая колонка = Left sidebar;
- Выделенное содержимое = Highlighted content;
- Содержимое = Content — основное содержимое страницы выводится сюда;
- Правая колонка = Right sidebar;
- Подвал = Footer;
- Помощь = Help

В нашей теме мы можем создавать любое количество областей. Если мы хотим изменить стандартный список, необходимо ввести его целиком, а потом дополнять или изменять. Стандартный список в формате info-файла:

```
regions[sidebar_first] = Left sidebar
regions[header] = Header
regions[highlight] = Highlighted content
regions[content] = Content
regions[help] = Help
regions[footer] = Footer
regions[sidebar_second] = Right sidebar
```

В скобках указывается название переменной, а справа — полное название в интерфейсе (в русском интерфейсе Drupal 6 его перевели на русский язык, что несколько некорректно). Лучше этот список только дополнять, чтобы не было проблем с совместимостью. Сделаем себе три подвала и отдельную область в правом верхнем углу. Вообще области нужны в тех местах, куда мы хотим выводить разную информацию. Наш файл в результате получился следующим:

```
name = SQ task control
description = Base theme for SQ site.
version = 1.0
core = 7.x
```

```
engine = phptemplate
stylesheets[all][] = drupal_fix.css
stylesheets[all][] = style.css
scripts[] = mm_images.js
regions[sidebar_first] = Left sidebar
regions[header] = Header
regions[righttop] = Right top corner
regions[highlight] = Highlighted content
regions[content] = Content
regions[help] = Help
regions[footer1] = Footer top 1
regions[footer2] = Footer top 2
regions[footer] = Footer
regions[sidebar_second] = Right sidebar
```

На этом мы закончим работу с .info-файлом и перейдем к файлу page.tpl.php. Этот файл служит для вывода HTML-страницы. В момент его вызова определены следующие группы переменных языка PHP.

Тип	Способ задания	Пример вывода внутри page.tpl.php
содержимое регионов	в скобках в .info-файле	`<?php print render($page['content']); ?>`

содержимое блоков	в зависимости от типа блока	`<?php if ($primary_nav): ?>`
		`...`
		`<?php print $primary_nav; ?>`
		`...`
		`<?php endif; ?>`

настройки темы и сайта	список элементов в .info-файле	`<?php if ($logo): print $logo; endif; ?>`
различные переменные	предопределенный список имен	`<?php print $breadcrumb; ?>`

Полный список переменных приведен в Приложении для дизайнера. Задача файла page.tpl.php — сделать выходную страницу на основе этих переменных. Простейший вариант создания такого файла:

```
<?php
// $Id$
?>
  <?php print render($page['header']); ?>
  <?php print render($page['sidebar_first']); ?>
  <?php print render($page['highlight']); ?>
  <?php print render($page['content']); ?>
  <?php print render($page['sidebar_second']); ?>
  <?php print render($page['footer']); ?>
  <?php print render($page['content']); ?>
```

Мы достаточно легко создали свой первый шаблон. Скопировали разметку предыдущего сайта внутрь `body` и разместили свои регионы в нужных местах внутри нее. Получился работающий шаблон, который можно постепенно дорабатывать и улучшать (до бесконечности). Список всех переменных находится на сайте для разработчиков: http://api.drupal.org.

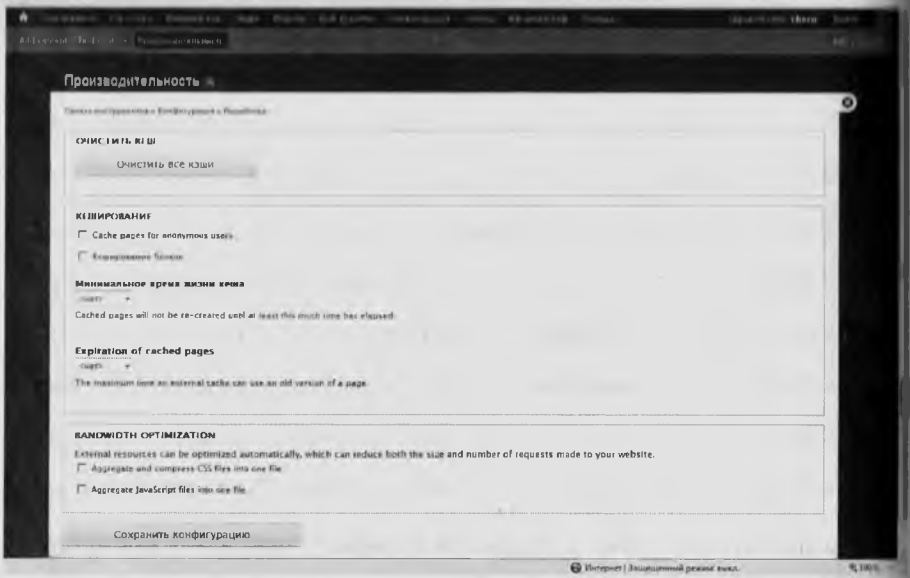
Рекомендуем изучить стандартные шаблоны. В них можно найти несколько типичных примеров.

Например, не очень удобно работать с картинками. Их достаточно просто вставлять через стили, но при вставке напрямую необходимо использовать выражение `base_path() . path_to_theme()` для формирования пути к директории темы.

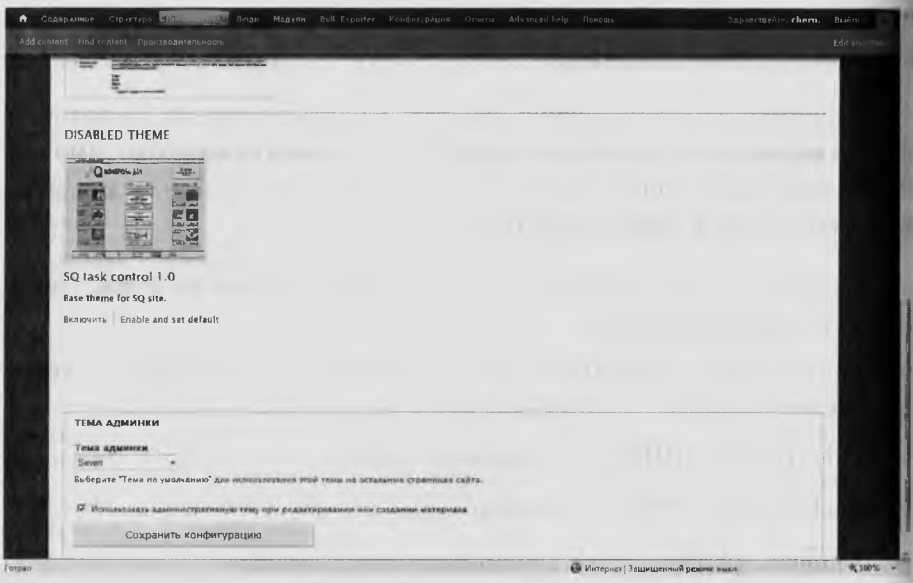
Запомним сразу еще ряд стандартных названий файлов темы, чтобы знать, что куда складывать:

- template.php — файл с нашими PHP-функциями; все функции, которые мы используем в теме, кладем сюда;
- node.tpl.php — PHP для показа материалов;
- block.tpl.php — PHP для показа блоков;
- comment.tpl.php — PHP для показа комментариев.

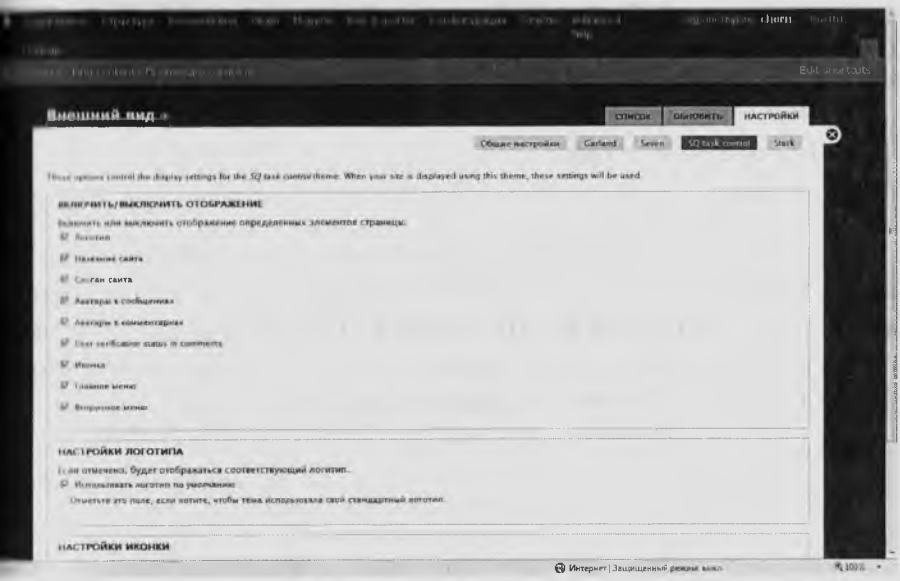
Да, чуть не забыли — нужно поместить директорию с нашей темой на сайт в директорию sites/all/themes. Затем необходимо очистить кэш. Помните, мы добавили в меню пункт **Производительность**?



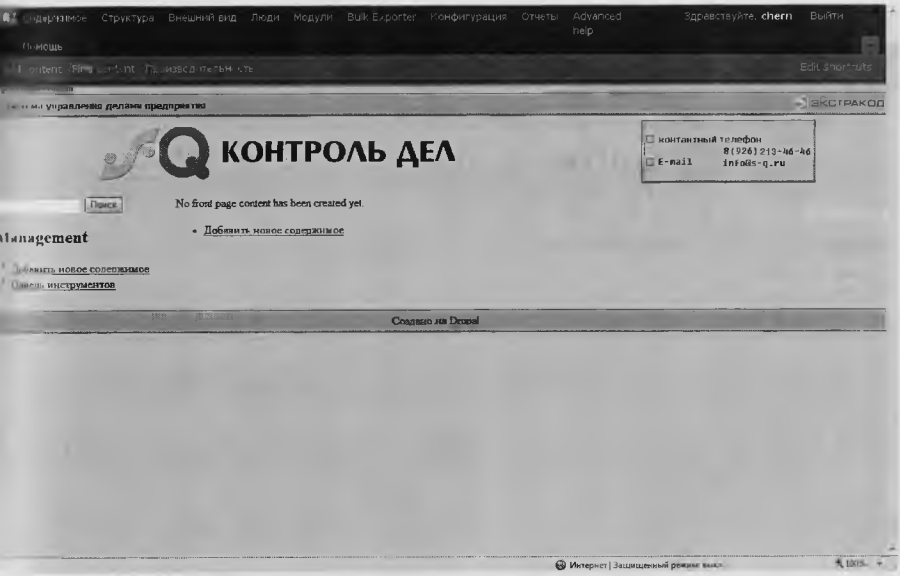
Открываем его и щелкаем по кнопке **Очистить все кэши**. После это го переходим в окно **Внешний вид**, активируем новую тему и ставим ей по умолчанию.



Также мы можем настроить тему — указать иконку, другой логотип и настройки сайта, которые хотим отображать.



Отметим, что Drupal буферизует .info-файл. Поэтому, если мы его ме-
ием, надо выполнить команду **Очистить все кэши** в меню **Производитель-
ость**. А пока закроем меню администрирования и посмотрим результат.



Верхнее меню почему-то потеряло стили. Иногда это происходит
з-за конфликтов стандартных стилей Drupal с нашими. Чтобы это по-

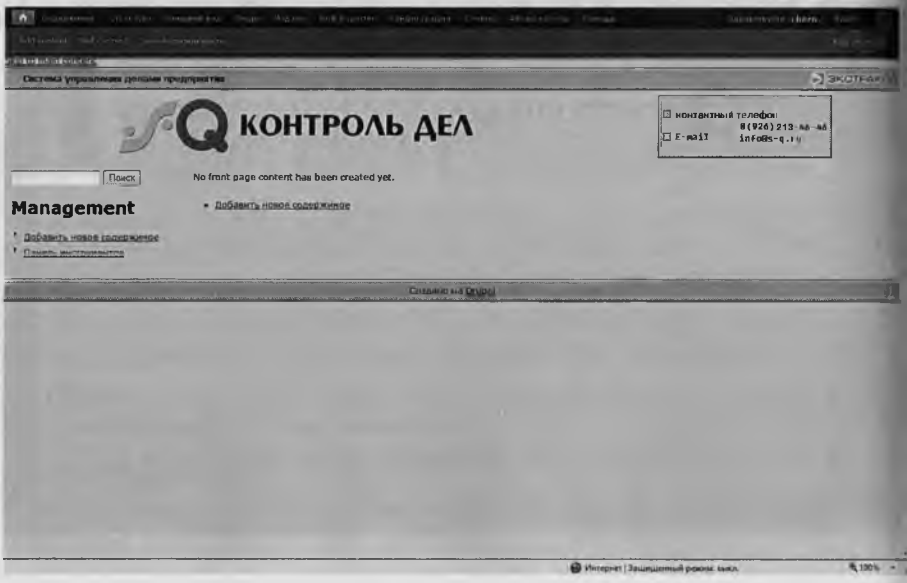
править, создаем файл drupal_fix.css и добавляем его в .info-файл. В не будут проводиться коррекции стандартных стилей, например:

```
tbody {
  border-top: 0px;
}
```

Их мы будем подбирать, пока разметка не станет приемлемой. Од нако в нашем случае все еще хуже. Оказывается, что стили для верхнего меню лежат среди стилей темы **Garland**. Поэтому нам придется само стоятельно установить стили для меню администратора, чтобы буковки были поменьше. Для этого добавим в drupal_fix.css стили:

```
.toolbar-menu {
  font: 76%/170% Verdana, sans-serif;
}

.toolbar-drawer {
  font: 76%/170% Verdana, sans-serif;
}
```

И проверим результат.



Все получилось!

ПРИМЕЧАНИЕ

Приведем также пример шаблона для вставки центральных областей страницы:

```
<table border="0" width="100%" cellspacing="10"
cellpadding="0">
  <tr>
    <?php if ($page['sidebar_first']): ?>
      <td width="20%" valign="top">
        <?php print render($page['sidebar_first']); ?>
      </td>
    <?php endif; ?>
    <td width="*" valign="top">
      <?php print render($page['highlight']); ?>
      <?php print render($page['content']); ?>
    </td>
    <?php if ($page['sidebar_second']): ?>
      <td width="20%" valign="top">
        <?php print render($page['sidebar_second']);
?>
      </td>
    <?php endif; ?>
  </tr>
</table>
```

ФОРМАТИРОВАНИЕ ЭЛЕМЕНТОВ

- Цепочки для вывода информации
- Обрамление материалов
- Форматирование View
- Форматирование блоков
- Показ фиксированных надписей
- Функция `t()`
- Перевод интерфейса

Итак, у нас складывается несколько стандартных цепочек для вывода информации:

- **Материал** ⇒ **Регион Content** ⇒ **Страница**;
- **Материал** ⇒ **View** ⇒ **Регион Content** ⇒ **Страница**;
- **Модуль** ⇒ **Блок** ⇒ **Регион** ⇒ **Страница**.

Давайте разберемся, как осуществляется настройка форматирования в каждом случае.

Обрамление материалов

Это просто показ материала на странице. Сам материал может содержать выделения слов, разбивку на абзацы, списки и т. д. Для этого мы пишем материал в форме Full HTML. Материал попадает в регион **Content**, который входит в страницу. Однако разметка может окружать материал, прежде чем он будет вставлен в регион. Такая разметка должна помещаться в файл node.tpl.php.

Приведем пример простого файла node.tpl.php:

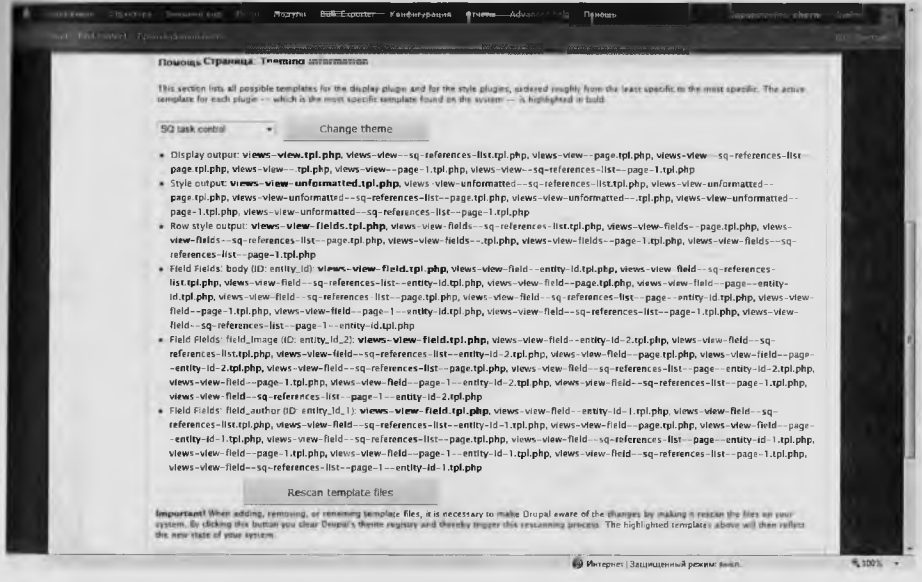
```
<?php
// $Id$

?>

<?php print render($content); ?>
```

В данном случае мы выводим только основное содержимое материала. Drupal по умолчанию выводит, например, информацию о публикации материала, что обычно нам совсем не нужно. Отметим, что данный файл используется для форматирования всех материалов сайта.

В файле мы можем использовать различные элементы. Их список приведен в конце книги в разделе приложений. Чтобы провести специфическое форматирование различных страниц, можно накладывать различные условия. Наша задача, впрочем, — задать форматирование вывода материала. Иногда интересно показать материал, в котором много полей. Для этого часто удобнее работать через модуль **View**.

Форматирование View

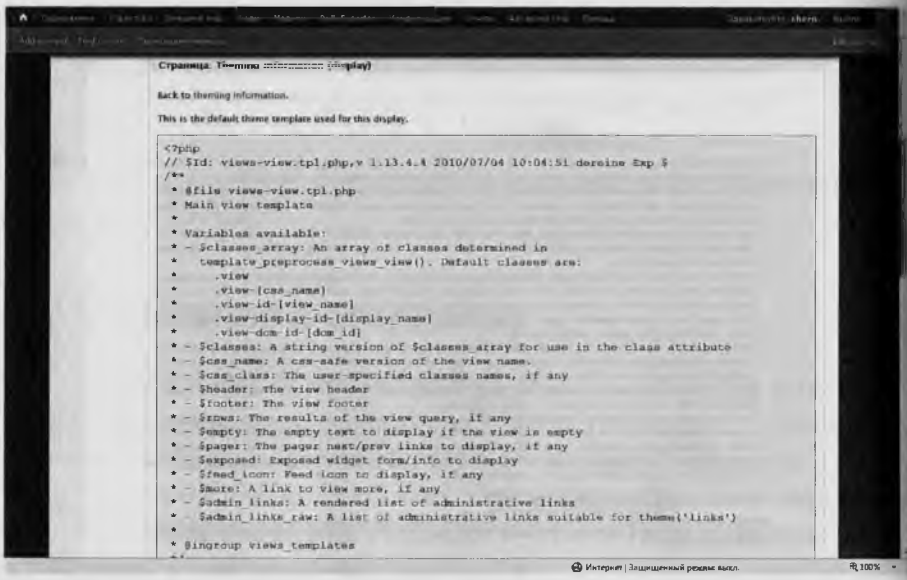
В данном случае нам нужно использовать аналогичные файлы шаблонов, чтобы изменить форматирование результата выполнения **View**. Чтобы узнать, какие файлы нам необходимы, зайдем во **View** и вызовем пункт **Theme:Information** в группе **Style settings**. Давайте посмотрим, что нам предлагают для **Страницы**.



Здесь указан список файлов шаблонов, которые мы можем пере-
пределить. Скажем, для показа страницы предлагается:

- views-view.tpl.php — общий шаблон для всех View. Выделен, по-
скольку сейчас используется;
- views-view--sq-references-list.tpl.php — шаблон для нашего View
sq_references_list;
- views-view--page.tpl.php — шаблон для всех показов страниц чере-
View;
- views-view--sq-references-list--page.tpl.php — шаблон для нашего
View в режиме **Страница**;
- views-view--page-1.tpl.php — шаблон для всех показов первой стра-
ницы через View — если показ по стран ицам;
- views-view--sq-references-list--page-1.tpl.php — шаблон для показа
первой страницы нашего View (если показ осуществляется по стра-
ницам).

Чтобы узнать, как получить доступ к различным элементам, нужно
или прочитать документацию, или пройти по ссылке и посмотреть теку-
щий шаблон показа. Примерно понятно, что и как использовать. Допу-
стим, мы хотим вывести текст отзыва и колонку в два столбца, а потом
поле с подписью. Необходимо использовать шаблон views-view--sq-ref-
erences-list.tpl.php. Щелкаем по кнопке **Display output**.



Рекомендуется просмотреть файлы на всех уровнях:

- **Display output** — формирует общий вид View. В качестве основы использует собранную переменную со строками (`$rows`);
- **Style output** — выводит строки в цикле, окружая их классами стилей. Используемый файл зависит от стиля вывода View;
- **Row style output** — выводит строку, собирая ее из полей;
- **Field...** — выводит поля. По умолчанию выводятся без форматирования.

Поскольку мы хотим расположить поля по-другому, создадим свой файл views-view-fields--sq-references-list.tpl.php и скопируем в него то, что было показано по ссылке **Row style output**. Затем поместим его в директорию с темой. Теперь снова перейдем в **theming information** и после щелчка по строке **Rescan template files** увидим, что выделен другой файл шаблона, то есть система подхватила этот файл. Щелкать по строке **Rescan template files** необходимо при создании нового файла шаблона — изменения старых файлов Drupal подхватывает на лету.

Теперь отредактируем наш файл. Кстати, в нем приведен список переменных, которые можно использовать. Не удаляйте его — может пригодиться.

Мы хотим вывести содержание и картинку автора в первой строке, а подпись — во второй. Потом выведем строку прочерка. Сделаем простой вариант:

```
<?php
// $Id$
/**
 * @file views-view-fields.tpl.php
 * Default simple view template to all the fields as a row.
 *
 * - $view: The view in use.
 * - $fields: an array of $field objects. Each one contains:
 *   - $field->content: The output of the field.
 *   - $field->raw: The raw data for the field, if it exists. This is NOT output safe.
```

```
    *   - $field->class: The safe class id to use.
    *   - $field->handler: The Views field handler
object controlling this field. Do not use
    *     var_export to dump this object, as it can't
handle the recursion.
    *   - $field->inline: Whether or not the field
should be inline.
    *   - $field->inline_html: either div or span
based on the above flag.
    *   - $field->separator: an optional separator
that may appear before a field.
    * - $row: The raw result object from the query,
with all data it fetched.
    *
    * @ingroup views_templates
    */
   ?>
   <tr>
     <td width=80% valign="top" align="left">
       <p align="justify"><font face="Tahoma"
size=2><?php print $fields[body]->content; ?></font>
     </td>
     <td width=20% valign="top">
       <p align="center"><?php print $fields[field_
image]->content; ?></p>
     </td>
   </tr>
   <tr>
     <td width=100% colspan=2 valign="top">
       <p align="right"><font size=2
face="Tahoma"><b><?php print $fields[field_author]-
>content; ?></b></font>
     </td>
   </tr>
   <tr>
```

```
    <td width=100% colspan=2 valign="top">
      <hr size=1>
    </td>
  </tr>
```

Отметим, что мы обращаемся к элементам массива `$fields` по идентификаторам, которые указаны на страничке **Theme:Information**. Вывод просто и понятно. Создайте только окружающие теги для таблицы в файле views-view--sq-references-list.tpl.php.

Итак, у нас получается иерархия для форматирования материалов.

Общее форматирование страницы	Файл page.tpl.php	Создаем внешний вид страницы и располагаем на ней наши регионы
Форматирование Node или View	node.tpl.php или views-view.tpl.php	Форматирование блока внутри страницы — название блока, специфические картинки и т. п.
Форматирование строк	views-view-fields.tpl.php	Форматирование размещения полей внутри View, а также, возможно, общие стили для полей — выделение, размещение
Форматирование текста	HTML внутри материала	Выделение слов, переносы строк, разбивка на абзацы, помещение внутритекстовых картинок

Форматирование блоков

Еще один вариант — показ блока из модуля. Это блоки типа **Поиск**, **FAQ**, **Опросы** и т. д. Мы сменим оформление блока поиска.

Сначала идем в директорию, где установлен модуль поиска. Этот стандартный модуль расположен в директории modules/search. Смотрим, какие в нем есть файлы шаблонов:

- search-block-form.tpl.php — шаблон для вывода блока поиска;
- search-result.tpl.php — шаблон для показа одного результата поиска;
- search-results.tpl.php — шаблон для показа списка результатов поиска.

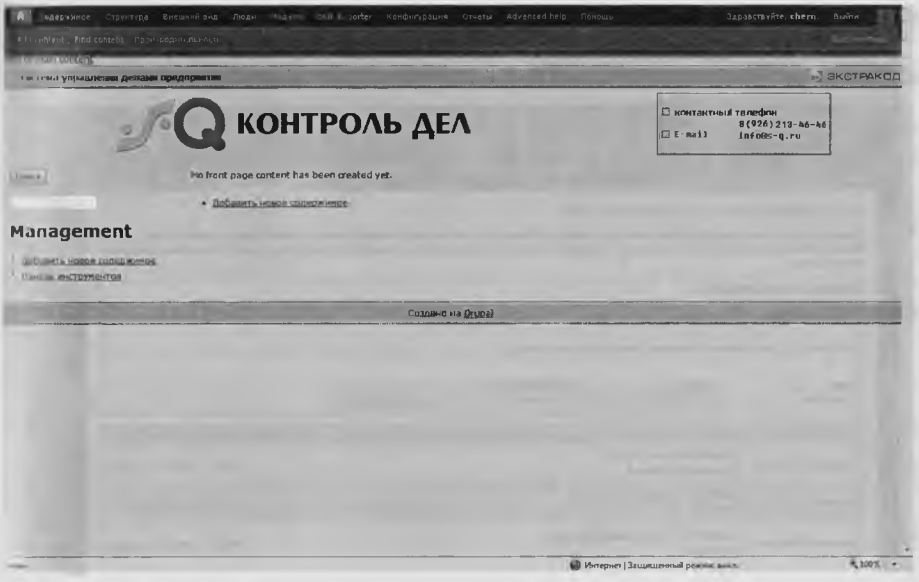
Поменяем шаблон показа блока. Для этого скопируем файл search-block-form.tpl.php в директорию с нашей темой и изменим его по своему

усмотрению. Наш файл будет главнее файла в директории модуля. Напишем в него что-нибудь для примера:

```
<?php
// $Id$

/**
 * @file search-block-form.tpl.php
 * Default theme implementation for displaying a
search form within a block region.
 *
 * Available variables:
 * - $search_form: The complete search form ready
for print.
 * - $search: Array of keyed search elements. Can
be used to print each form
 *   element separately.
 *
 * Default keys within $search:
 * - $search['search_block_form']: Text input area
wrapped in a div.
 * - $search['actions']: Form submit button.
 * - $search['hidden']: Hidden form elements. Used
to validate forms when submitted.
 *
 * Since $search is keyed, a direct print of the
form element is possible.
 * Modules can add to the search form so it is
recommended to check for their
 * existance before printing. The default keys will
always exist.
 *
 *   <?php if (isset($search['extra_field'])): ?>
 *     <div class="extra-field">
 *       <?php print $search['extra_field']; ?>
```

```
 *      </div>
 *    <?php endif; ?>
 *
 * To check for all available data within $search,
the code below.
 *
 *    <?php print '<pre>'. check_
plain(print_r($search, 1)) .'</pre>'; ?>
 *
 * @see template_preprocess_search_block_form()
 */
?>
<?php print $search['actions']; ?>
<?php print $search['search_block_form']; ?>
```

Поместим блок поиска в левую колонку, очистим кэш и убедимся, что форма изменилась.



Иногда для изменения внешнего вида необходимо писать специальную программу и нельзя обойтись простыми настройками шаблона.

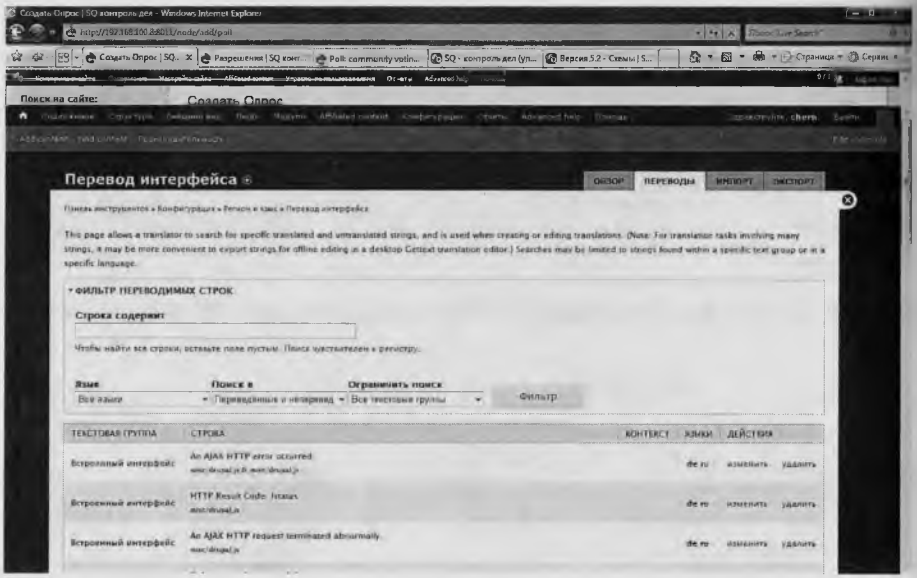
Показ фиксированных надписей

Иногда на сайте нужно показать фиксированные надписи, например телефонные номера на первой странице и т. д. Это особый тип материала, который мы хотим вывести в блоке. Проще всего это сделать, построив View типа блок. В этом случае вы сможете менять все эти надписи «правильно» — изменяя содержимое материала.

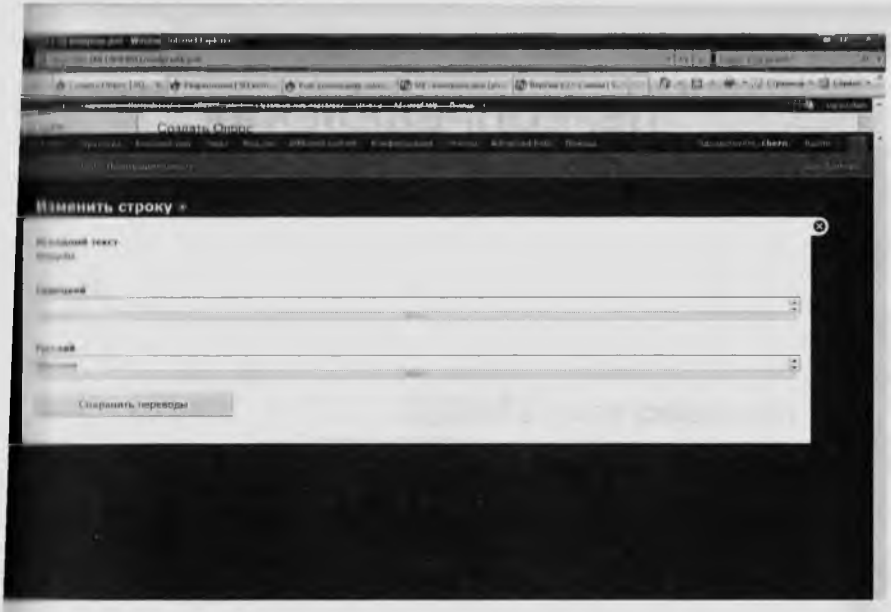
Второй способ вывода фиксированных надписей основан на функции `t()`. Она служит для показа фиксированных надписей на языке сайта. Допустим, на странице присутствует кнопка поиска. На русском языке она должна называться **Искать**, а на английском — **Go**. Для вывода подобных надписей и служит функция `t()`. Использовать ее очень просто — в tpl.php вводим вывод в виде:

```
<?php print t('english message'); ?>
```

Потом открываем страницу с этой надписью. При этом *english message* будет занесено в базу. Перевести его на все языки можно на странице **К ⇒ Перевод и язык ⇒ Перевод интерфейса ⇒ Переводы**.



Нужно выполнить поиск исходной строки на английском языке, после чего ввести ее перевод на русский и другие языки.

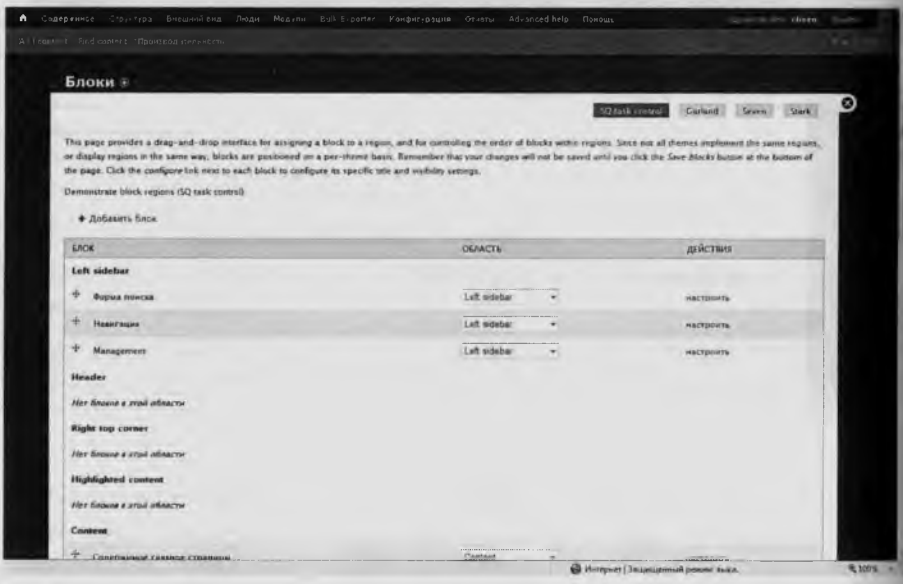


После этого показ строки будет определяться выбором основного

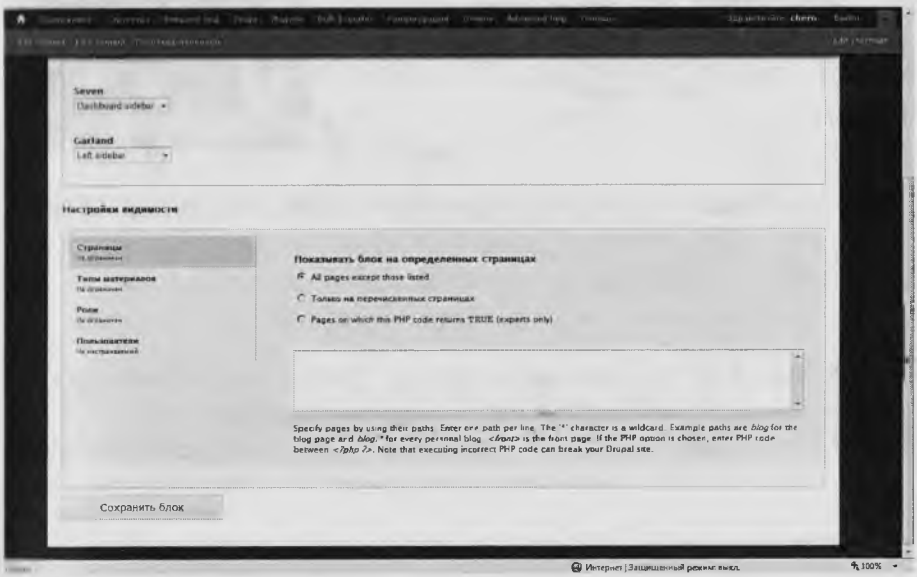
ПЕРЕОПРЕДЕЛЕНИЕ ШАБЛОНОВ

- Настройка показа блоков
- Devel и Theme developer
- Шаблоны-кандидаты для страницы

Мы можем задать внешний вид страницы и регулировать ее содержимое путем указания списка страниц, на которых должны отображаться блоки. Это делается через меню **Структура** ⇒ **Блоки**.



Щелкнув по кнопке **Настроить**, мы сможем указать список страниц, на которых необходимо выводить блок, и некоторые другие опции.

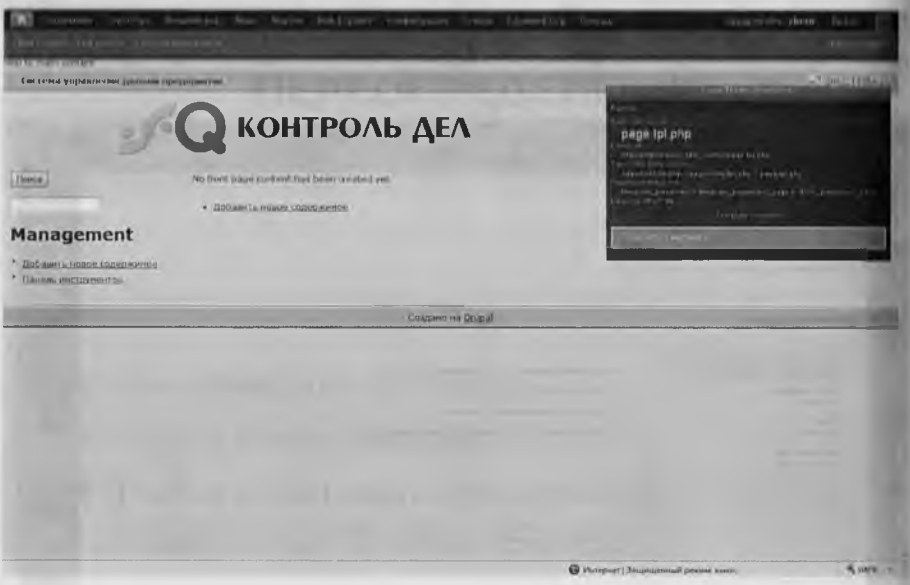


Например, для вывода списка отзывов можно использовать еще один вид показа — **Блок**, поместив его на определенные страницы. Кстати, **Содержимое главной страницы** — это тоже один из блоков, и мы можем расположить его в любом месте.

Также (при желании оформить разные страницы по-разному) можно регулировать внешнюю HTML-разметку при помощи условных конструкций в файлах tpl.php. Однако, если страниц много, условные конструкции будут слишком сложными и непонятными. Чтобы избежать этого, следует использовать механизм переопределения шаблонов. Этот механизм идентичен переопределению шаблонов View.

Для начала необходимо установить полезный вспомогательный модуль **Devel** (http://drupal.org/project/devel). Модуль служит для помощи в разработке, нам нужны два его компонента: сам **Devel** и **Theme Developer**. После установки на экране отобразится маленькое окошко **Themer info**. Поставив флажок, мы увидим большое окно с просьбой указать элемент на экране для показа информации о нем.

Допустим, мы включим данный модуль при показе окна с темой сайта и укажем логотип. Получится приблизительно следующая картина.



Сейчас нам наиболее интересен список с кандидатами шаблонов:

```
page--front.tpl.php < page--node.tpl.php < page.
tpl.php
```

Здесь выводится список шаблонов и указывается порядок, в котором они используются. Давайте разберемся с ним поподробнее.

Шаблон, который сейчас используется, называется page.tpl.php, и мы также можем увидеть путь к нему. Путь понятен — это шаблон, который мы создавали и сохраняли в директории своей темы. Мы видим, что сейчас действительно используется именно он.

Следующий кандидат — page--node.tpl.php. Это означает, что мы можем создать специальный шаблон для страницы конкретного материала. Если вы помните, все материалы в Drupal имеют свой идентификатор, и путь к ним по умолчанию — /node/id_node, а уже потом на него накладываются синонимы.

Мы создадим шаблон вида page--node--id_node.tpl.php, который будет использоваться только для страницы с этим материалом.

Не очень удобно, что нельзя использовать один подобный шаблон для нескольких страниц, зато можно сделать шаблон вида page--node.tpl.php, и он будет применяться ко всем страницам с материалами.

А, скажем, список отзывов в данном случае будет показан через основной шаблон page.tpl.php.

Еще один кандидат — это page--front.tpl.php. Он используется для показа главной страницы. Действительно, обычно главная страница сильно отличается от остальных. Она задается в меню **К ⇒ Системи ⇒ О сайте**. При показе Drupal проверяет шаблоны слева направо и, отыскав подходящий, использует его.

На самом деле структура наследования шаблонов более запутанная, чем показывает модуль **Theme Developer**. В Приложениях для дизайнера приведена полная структура наследования шаблонов. Однако для начинающих вполне достаточно простых вариантов переопределения.

Есть еще один вариант переопределения через функции вывода темы. Вариант с функциями требует программирования, а работа с шаблонами проще и нагляднее. Поэтому, если у вас не очень сложный сайт, проще использовать переопределение шаблонов.

Чтобы шаблон заработал, ОБЯЗАТЕЛЬНО должен быть переопределен базовый шаблон. То есть, чтобы заработал page--front.tpl.php или page--node.tpl.php, в теме НЕОБХОДИМО завести page.tpl.php. Также рекомендуется очистить кэш перед применением шаблонов.

Иногда переопределять приходится функции. Переопределяют функции, которые начинаются с префикса `theme_`. В нашей теме в файле template.php нужно завести аналогичную функцию, но с префиксом имени самой темы. Проще всего скопировать стандартную тему и править ее. Для этого в окошке модуля **Devel** нужно щелкнуть по имени функции, после чего мы попадем в документацию, где будет указано, откуда ее взять.

ПРИМЕЧАНИЕ

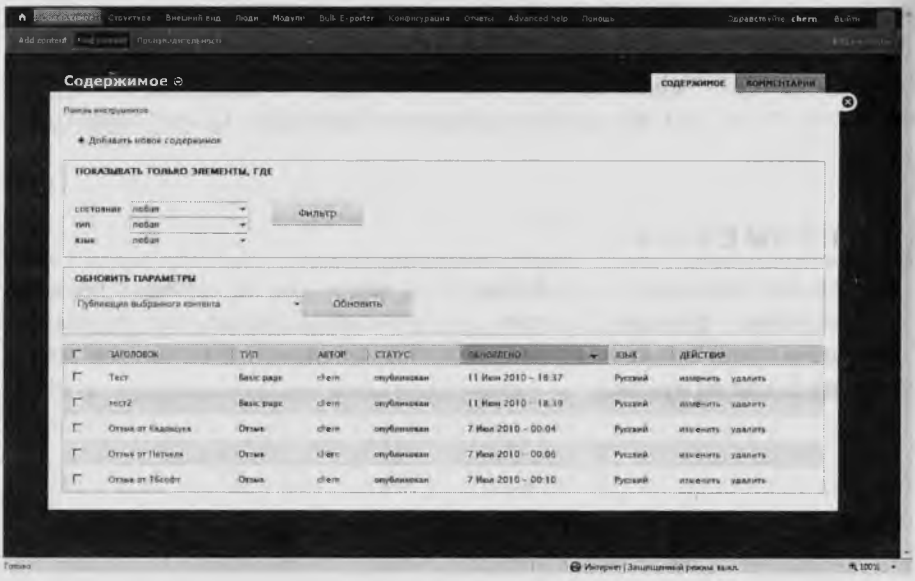
При включении модуля **Devel** у нас возникали проблемы с работой основных функций, поэтому рекомендуем включать его (путем выдачи прав) только на время настройки шаблонов. В остальное время он особо не нужен.

ПОКАЗ СЛОЖНОГО МАТЕРИАЛА

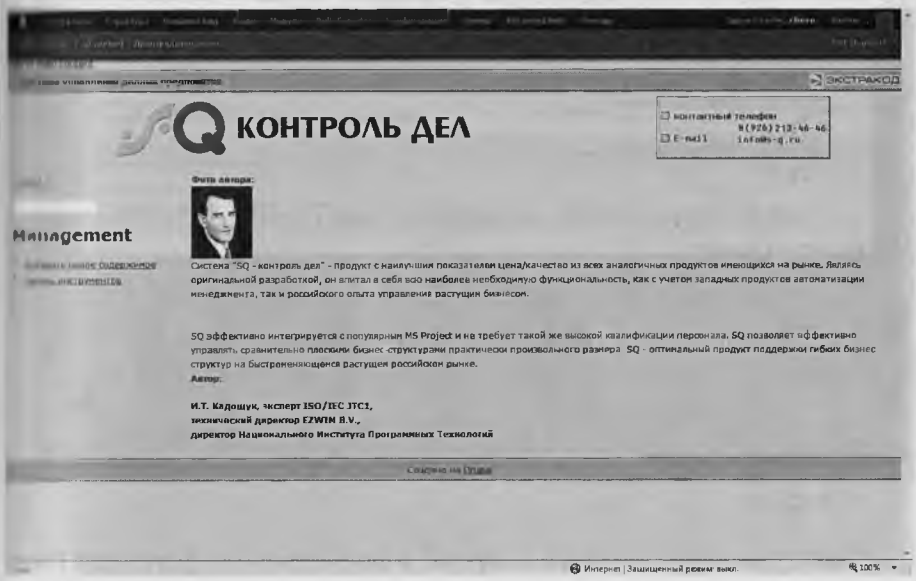
- Поиск и просмотр материалов
- node--type.tpl.php
- Управление отображением полей
- Оформление центральной страницы

Под сложным материалом в данном случае мы понимаем такой материал, который кроме названия и содержания имеет еще дополнительные поля. Это, например, материал типа **Отзыв**, с которым мы работали. Для нашего сайта мы сделали новый тип материалов, к каждому из которых привязана картинка и дополнительное краткое название. Показ таких материалов оформляется через шаблон node--type.tpl.php.

Давайте посмотрим, как сейчас выглядит отзыв, если его просматривать в виде страницы. Для этого зайдем в список материалов, которые ввели на сайт — **Find content** (Поиск содержимого).



Здесь можно отбирать материалы по различным параметрам, а также переходить на страницу того или иного материала. Также можно выполнять определенные действия над группой материалов, например публиковать или прятать их. Выберем один из материалов и посмотрим, как он будет выглядеть в виде страницы.



Мы видим, что поля следуют одно за другим с названиями, которые мы им присвоили. Давайте отформатируем свой отзыв так, чтобы он выглядел более прилично.

Нам нужно работать с файлом node--reference.tpl.php. Заведем его в теме и введем туда собственную HTML-разметку. Для вывода содержимого определенного поля надо писать:

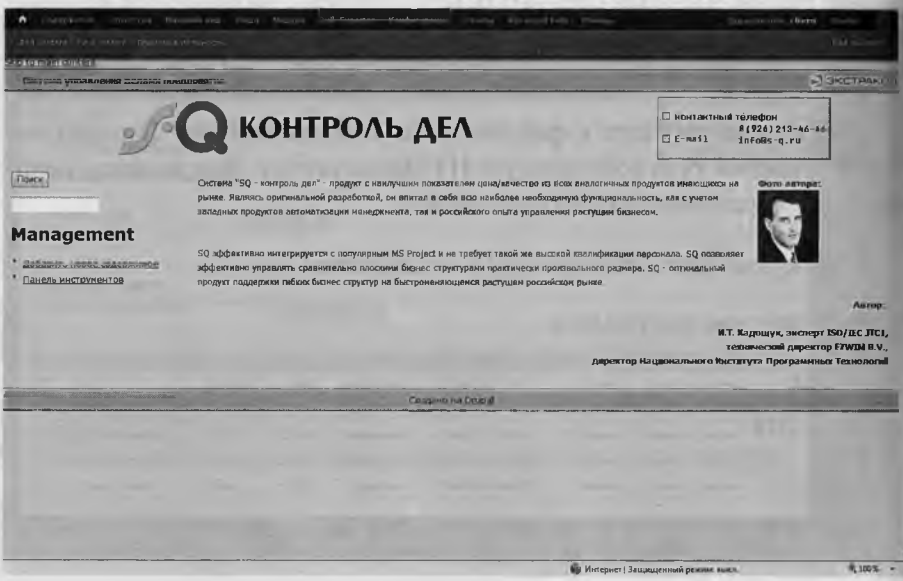
```
<?php render($content['field_name']); ?>
```

Вот что у нас получилось:

```
<?php
// $Id$
?>

<table border="0" width="100%" cellpadding="6">
  <tr>
```

```
      <td width=80% valign="top" align="left">
        <p align="justify"><font face="Tahoma"
size=2><?php print render($content['body']); ?></font
      </td>
      <td width=20% valign="top">
        <p align="center"><?php print
render($content['field_image']); ?></p>
      </td>
    </tr>
    <tr>
      <td width=100% colspan=2 valign="top"
align="right">
        <p align="right"><font size=2
face="Tahoma"><b><?php print render($content['field_
author']); ?></b></font>
      </td>
    </tr>
  </table>
```

Скопируем данный файл в Drupal и посмотрим на результат (возможно, понадобится стереть кэш).

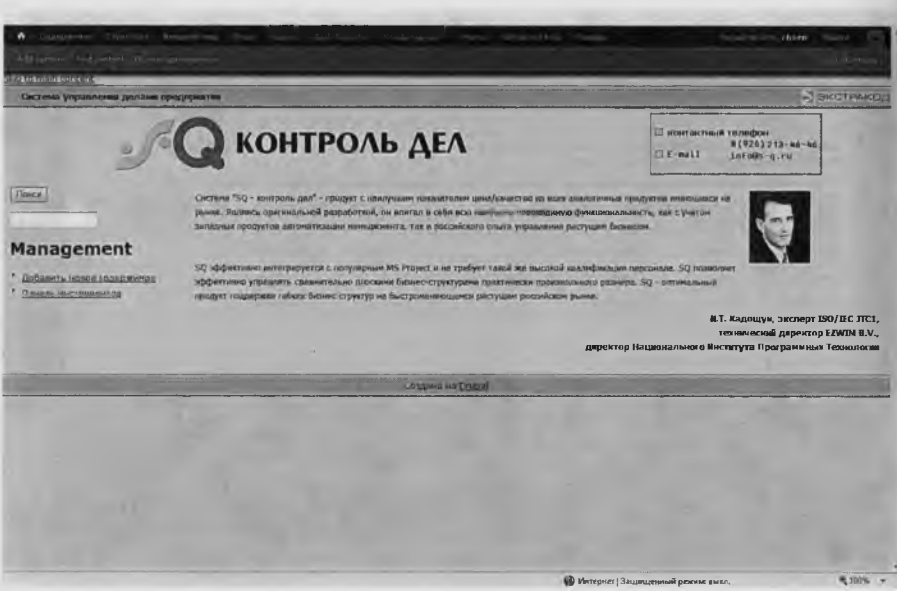


Все хорошо, только заголовки полей лишние. Но у нас есть хитрая опция. Давайте заглянем в меню **Структура** ⇒ **Типы материалов**.

Здесь напротив каждого из типов стоит линк **управлять отображением**. Откроем его для типа **Отзыв**.

В этой табличке мы можем настроить, каким образом должны ото бражаться поля материала в двух режимах: в режиме аннотации (мы см дойдем до этого) и в режиме полного просмотра (FULL CONTENT Мы можем настроить наличие и расположение заголовка поля. Давай уберем метки для всех полей.

Сохраняем изменения и смотрим на результат.



Ура! Мы получили отзыв «с человеческим лицом», извините за каламбур.

Все остальные опции, которыми можно воспользоваться в файле node.tpl.php, приведены в Приложении для дизайнера.

Оформление центральной страницы мы тоже сначала хотели реализовать через специальный тип материала. Вообще для этого существуют три способа.

1. Самый примитивный. Сложить внутреннее содержимое центральной страницы в материал типа **Page** и опубликовать его на центральной странице. Тоже возможно, но менее формализовано, поскольку нам нужно будет всю сложную разметку поместить внутрь материала. А потом могут возникнуть проблемы с шаблонами показа типа **Page** и шаблона центральной страницы.
2. Создать отдельный тип материала для центральной страницы и настраивать показ страницы через node-type.tpl.php. Это мне больше импонирует, поскольку можно завести формализованные поля для центральной страницы. При этом немного проще управлять разметкой внутри файла шаблона.
3. Сформировать отдельный page-front-tpl.php. Это повлечет за собой дублирование общей разметки с существующим page-tpl.php. Делать отдельный шаблон для первой страницы нужно, если она принципиально отличается от других страниц. На самом деле этот способ зачастую самый разумный, и именно его следует применять.

И какой из этого следует вывод? А такой, что для решения одной и той же задачи в Drupal всегда найдется несколько способов. При этом каждый раз надо искать компромисс между правильностью с точки зрения Drupal, простотой реализации и легкостью дальнейшей поддержки.

МЕНЮ РАЗНЫХ УРОВНЕЙ

- Виды меню в Drupal
- Создание меню
- Показ меню в виде блока
- Настройка внешнего вида меню
- Ручной показ меню
- Jump и Nice menus
- Показ всплывающей подсказки

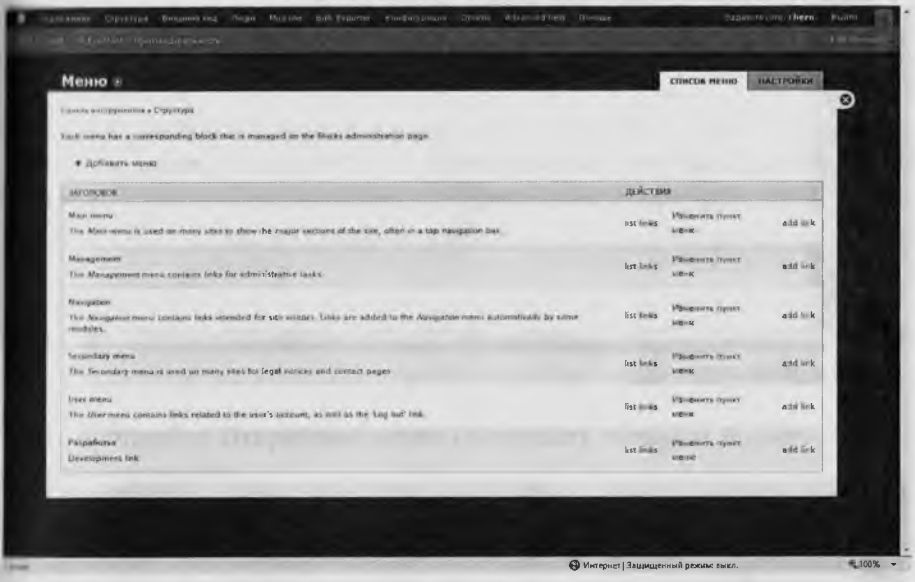
Итак, мы научились создавать и оформлять страницы. Нам осталось добавить на сайт навигацию.

На старом сайте (www.s-q.ru) присутствовало несколько видов меню:

- основное меню — список основных страниц сайта в виде раскрывающегося списка, размещается в блоке справа вверху;
- дополнительное меню — список из трех страниц в подвале сайта;
- меню главной страницы — специального вида кнопки в основной части сайта (ведут на страницы из основного меню).

Теперь надо разобраться, как все это организовать с точки зрения Drupal.

Давайте разберемся, как создавать и использовать различные виды меню, проследовав в подменю **Структура ⇒ Меню**.



Здесь описаны шесть стандартных меню:

1. **Main menu** — данное меню используется для показа основного навигационного меню сайта. Может быть показано как закладки в верхней части сайта. Это основное меню сайта, которое создается в первую очередь.

2. **Management** — меню для администрирования. Это верхняя строка и все, что под ней. Мы его не трогаем — данное меню формирует Drupal.

3. **Navigation** — меню для посетителей сайта, но основанное на функциях подключенных модулей. Тоже не будем его пока трогать.

4. **Secondary menu** — данное меню обычно используется для навигации по таким страницам, как **Правовая информация**, **Контакты** и другая вторичная навигация второго уровня важности. Данное меню обычно располагается внизу сайта, и с ним мы будем активно работать.

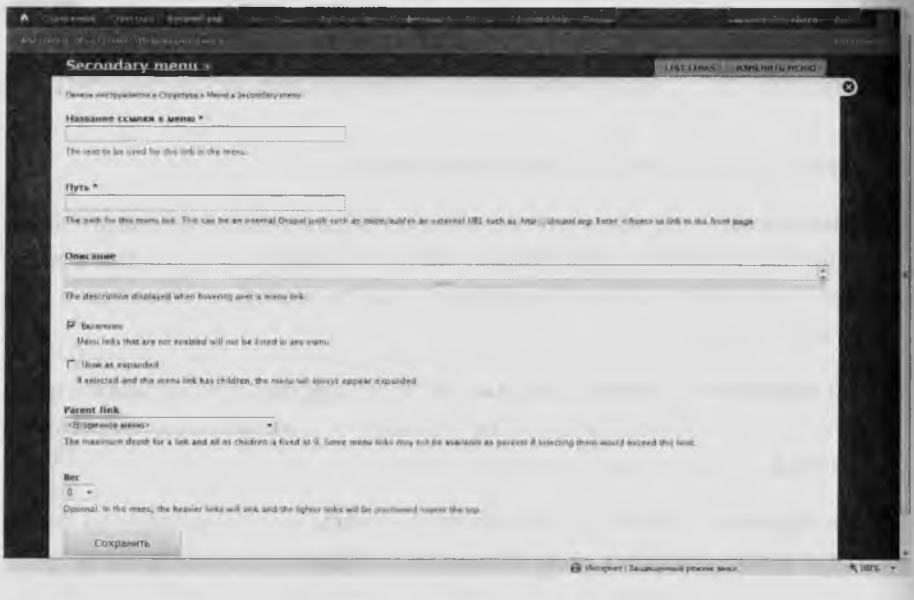
5. **User menu** — меню для линков, связанных с учетной записью пользователя, например, для просмотра своих данных или выхода из системы.

   На данный момент в нем изначально созданы два пункта, которые используются в том числе административной темой. Мы можем использовать это меню по своему усмотрению.

6. **Разработка** — линки для разработки. Не трогаем.

Вроде бы, все понятно — надо описать наше **Основное меню** как **Main menu**, **Дополнительное меню** — как **Secondary menu**, а на главной странице просто использовать пункты меню из основного меню по идентификатору и расположить там в любом порядке.

Вводить меню очень просто — для каждого пункта нужно указать ссылку, по которой осуществляется переход, название пункта и комментарий при наведении.



Создадим несколько пунктов на существующие страницы. Можно, кстати, делать ссылки и на внешние страницы. Теперь добавим блок нужного меню в то место, где хотим его увидеть.

1. «Как положено» — включаем **Theme developer** и начинаем изучать что и как надо переопределять:

   - функция `theme_menu_item_link` — формирует линк;
   - функция `theme_menu_item` — обвязывает линк тегами `<li>`;
   - функция `theme_menu_tree` — обвязывает список линков тегами `<ul>`;
   - шаблон `block.tpl.php` — добавляет заголовок и пару `<div>` для обрамления блока.

   Для создания своего вида меню нужно честно перехватить и переписать все эти шаблоны и функции. Жутко утомительное дело.

2. «На халяву» — этот метод, по сути, является одним из сниппетов (про сниппеты см. в Приложениях для разработчика). Мы не будем использовать показ меню как блока и уберем его из областей. Вместо этого в файл page.tpl.php мы вставим примерно такие строки:

```
<?php if (is_array($secondary_menu)) : ?>
<ul id="secondary-nav">
<?php foreach ($secondary_menu as $link): ?>
<li><?php

$href = $link['href'] == "<front>" ? base_path() :
base_path() . $link['href'];
print "<a href='" . $href . "'>" . $link['title'] .
"</a>";

?></li>
<?php endforeach; ?>
</ul>
<?php endif; ?>
```

Данный текст обрабатывает только внутренние линки. Для обработки внешних линков придется поставить еще одно условие. Зато тут очень легко ввести необходимый вам вид разметки. Жалко, что этот метод работает только для первичных и вторичных линков.

Вообще говоря, рисование меню — это всегда хлопотное дело. Обычно оно заключается в том, что скачивается некая стандартная библиотека обработки меню, после чего настройка идет, в основном, стилями.

Типичный пример такой библиотеки — модуль **Jump** (http://drupal.org/project/jump), который позволят создавать меню в виде раскрывающегося списка. Это очень простой модуль, на основе которого можно легко разобраться, как делать модули.

Встраивается этот блок очень интересно. Он автоматически порождает еще один блок для каждого меню в системе. После этого нужно лишь убрать блок обычного меню и вставить блок **Jump menu**. Отметим, что данный блок работает несколько кривовато и некорректно обрабатывает линки, когда мы находимся в URL с поддиректориями. Так что напишите пару строчек вместо него — не поленитесь.

Еще есть более мощный блок **Nice menus** (http://drupal.org/project/nice_menus), который позволяет создавать иерархические раскрывающиеся меню. Думаю, существует и множество других вариантов.

ПРИМЕЧАНИЕ

Есть забавный вопрос — все функции вывода меню используют название и линк. Однако для пункта меню можно задать и название, которое будет всплывать в результате наведения мышки (title). Его можно найти в `$link['attributes']['title']`.

ПРИМЕЧАНИЕ

При вводе материала мы можем сразу ввести пункт меню, где этот материал будет размещен. Естественно, для этого нам нужны правильные права.

ТАКСОНОМИЯ

- Что будем делать
- Что такое таксономия
- Словари и семантика
- Настройки термина
- Использование таксономии
- Отличия Drupal 7
- Замечания

Вот мы и добрались до таксономии — помните, это то слово что испугало автора в начале. На самом деле все очень просто. Или сложно — как посмотреть.

Проблема тут в том, что на момент написания книги таксономия в Drupal 7 практически отсутствует, поскольку это не самый важный блок. Поэтому мы построим данную главу следующим образом: опишем, что было в Drupal 6, а потом добавим то, что мы увидели в Drupal 7. Дальше вам придется разбираться самостоятельно. Никто не говорил, что все будет так просто...

Что же такое таксономия? Это возможность создания различных группировок материалов в соответствии со справочниками. Допустим, создаем каталог фильмов, и мне попадается неординарный фильм, например «Гостья из будущего». Это детский фильм, но он может классифицироваться и как фантастика. Кроме того, он нашего производства. Что же делать?

Таксономия позволяет создать множество различных классификаторов. Причем:

- число классификаторов не ограничено;
- мы можем отнести материал сразу к нескольким терминам (классам);
- термины могут создаваться на лету;
- можно создавать классификаторы для определенных типов материалов.

В дискретной математике известно понятие классификации. Но там необходимо все материалы разбить по конкретным классам.

В Drupal мы можем управлять разбиением по классам:

- задать типы материалов, которые классифицируем;
- задать возможность создания терминов пользователями;
- указать, можно ли относить один материал к множеству терминов-классов;
- определить, нужно ли при создании обязательно относить материал к одному из терминов-классов данного словаря.

Тут есть проблема в именовании. Например, словарь — это классификатор, а термин — класс. Термин еще путается с тегом. Видимо, русские названия пришли путем буквального перевода. Главное — понимать их смысл.

Мы покажем, как можно использовать классификаторы. Например, на нашем сайте все странички были раскрашены в желтый, синий или зеленый цвет. Классификатор ли это? Да, но он некоторым образом нарушает правила семантики.

Мы раскрашивали страницы в разные цвета в зависимости от того, к какой группе информации они относились:

- синий — общая информация о программе;
- желтый — информация о продаже, поддержке, лицензиях и т. п.;
- зеленый — клиентская и партнерская информация.

Соответственно, правильный классификатор — это тип информации! А цвет страницы можно задавать в зависимости от группы инфор-

мации для материала. Это будет правильно, и смысл перестанет зависеть от презентации.

Создадим классификатор по типу материала и раскрасим страницу в зависимости от него.

Перейдем в меню **Структура** ⇒ **Таксономия**. Укажем **Добавить словарь**. Здесь присваивается название, дается описание и указывается список типов материалов, для которых будет использоваться данный словарь. Понятно, что если терминология относится к фильмам, нет смысла вводить эти термины для **Отзывов**. А дальше мы как раз определяем, кто и каким образом может вводить новые термины.

Мы можем определить, что создали список (а пользователям остается только выбирать подходящие термины), можем указать, хотим ли, чтобы один материал относился к нескольким классам (например, детский и фантастика), можем потребовать обязательности ввода термина. Все зависит от вашей предметной области.

При добавлении словаря, как и при добавлении нового типа материала, возникают новые права на него, которые надо давать вручную через меню **Люди** ⇒ **Разрешения**.

Итак, мы ввели словарь. Теперь введем термины (классы). Открываем список и отмечаем опцию **Добавить термин**.

Название и описание понятны. Но здесь присутствуют и дополнительные настройки:

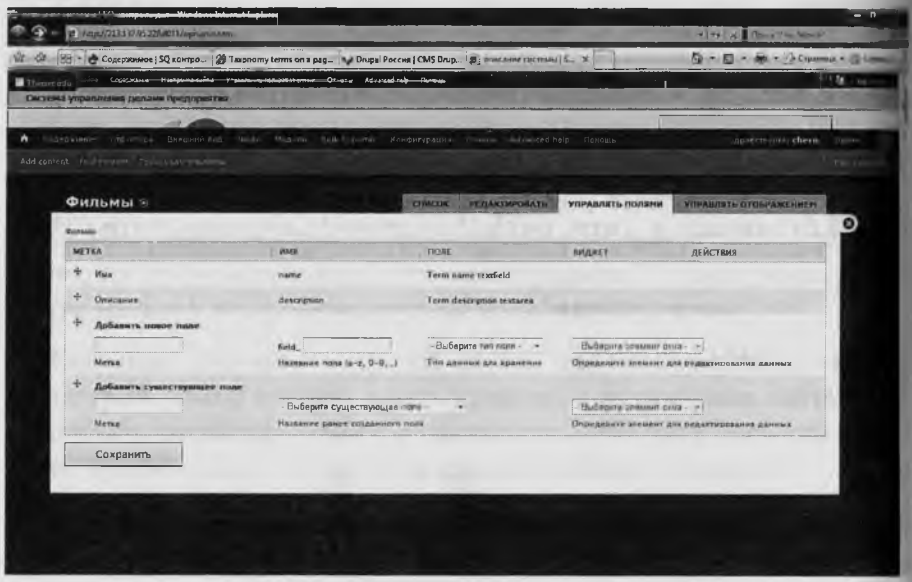
- **Родители** — в документации указано, что наличие родителя должно быть определено в описании словаря. Однако я не нашел этой опции. Поэтому словари всегда иерархические, что, в общем, не мешает привязывать материалы на любой уровень иерархии. Когда мы выбираем родителя, словарь организуется в иерархическом виде. Важно, что мы можем выбрать несколько родителей для термина (и это нормально), но, конечно, система не позволит циклов;
- **Связанные термины** — список служит для создания связанных категорий. Очень хорошо давать примеры с кулинарных сайтов. Допустим, китайская и тайская еда вполне могут быть связанными категориями. Потом посмотрим, как их можно использовать;
- **Синонимы** — синонимы нужны для решения спорных вопросов с русским языком (в первую очередь). Мы можем указать, что «важ-

После этого в переменной $color лежит нужный цвет. В зависимости от него форматируем табличку:

```
<?php if($color=='yellow'): ?>
  <td width=100% style="border: 1 solid
#000000" bgcolor=#EBF19D>
<?php elseif($color=='green'): ?>
  <td width=100% style="border: 1 solid
#000000" bgcolor=#B6EBC2>
<?php else: ?>
  <td width=100% style="border: 1 solid
#000000" bgcolor=#8BC6F8>
<?php endif; ?>
```

После этого, задавая признак тематики для нашего типа материала, мы получим различные цвета страницы.

А теперь давайте проговорим, что появилось в Drupal 7. Главное – это то, что термины таксономии теперь являются полноценными объектами и могут иметь дополнительные поля. Это означает, что мы можем не просто описывать термины (основу классификации), но и классифицировать их.



Например, ввести термин *режиссер*, который будет не только при-
ан к определенным фильмам, но и иметь имя, фотографию, биогра-
ию и т. п. В принципе такая структура находится в одном шаге до про-
модульного хранения контента в произвольно взаимосвязанном наборе
ущностей, но пока еще данный вопрос до конца не проработан. Есть
ишь выделенные типы, например пользователь или термин, которые
меют семантическую привязанность. В идеале, база для хранения дан-
ых — это база, а семантика — это другой уровень, который возникает,
огда мы начинаем показывать свои данные пользователю. Но –это, бу-
ем надеяться, дело дальнейших версий.

ПРИМЕЧАНИЕ

Для простых сайтов таксономия особо не нужна. Но если на вашем сайте множество материалов, то таксономия очень полезна для их группировки — именно правильная организация контента отличает хороший сайт от плохого. Особенно полезно то, что модуль **View** позволяет делать выборки на основе таксономии. Однако постройка правильной таксономии на сайтах с большим количеством материалов — это задача нетривиальная. Поэтому построению таксономии и ее поддержанию обязательно нужно уделять время — и оно не пропадет зря.

ПРИМЕЧАНИЕ

Не стоит делать таксономию более чем на 30–40 терминов. Иногда бывают исключения, например страны, которых более 250. По опыту техническая документация даже для очень больших систем не превышает 100 терминов.

НОВЫЕ ВОЗМОЖНОСТИ

- План на будущее
- Бэкап сайта
- Закладки на странице
- Новости
- Форум
- Внедрение специальных форм
- Опросы
- Книжки
- Статистика

В принципе мы уже изучили все, что нужно для переноса сайта со старой системы на новую. Осталось ввести информацию, создать главное меню, настроить оформление — в общем, выполнить чисто техническую работу, из которой и состоит 90% поддержки сайта. Однако у нас будет еще возможность позаниматься интересными вещами. Следующий пункт нашего плана — добавить новые возможности.

Это означает, что нужно найти девочку, посадить ее вбивать старую информацию, а самим заняться новой. Итак, чего же еще нового мы хотим:

Бэкап сайта

- Backup and Migrate
- Files
- Быстрый бэкап
- Директории для бэкапа
- Список выполненных бэкапов
- Настройка расписания
- Задание профиля
- Делайте тренировки

Для бэкапа необходимы два модуля:

- **Backup and Migrate** (http://drupal.org/project/backup_migrate) — бэкап базы;
- **Backup and Migrate Files** (http://drupal.org/project/backup_migrate_files) — плагин для бэкапа файлов; его пока не было для версии 7, но рассказывать мы про него можем...

Для миграции файлов нужен специальный модуль **PEAR**. О его инсталляции написано в README.txt для модуля **Backup and Migrate Files**. Без него нельзя будет бэкапить файлы.

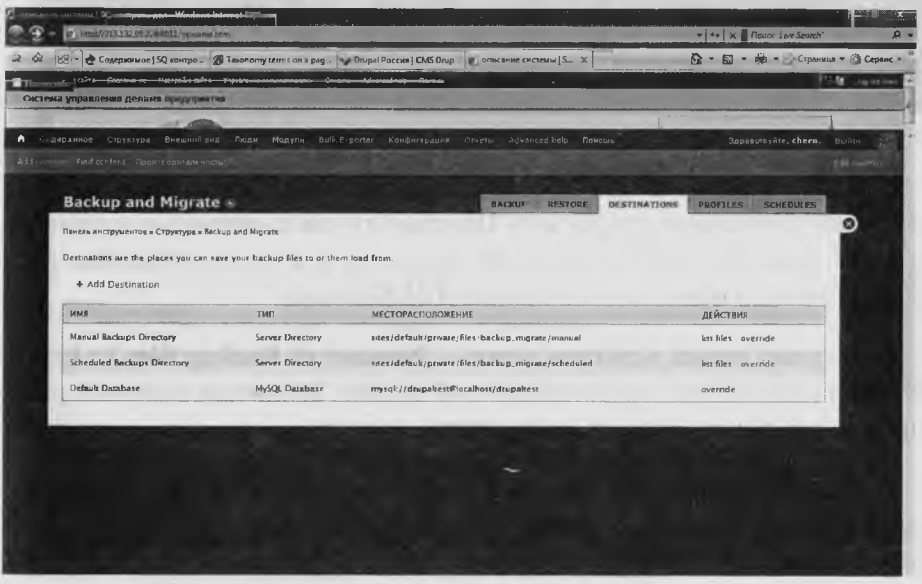
Как обычно, находим нужные файлы и устанавливаем их. В меню появляется новый пункт **Структура** ⇒ **Backup and Migrate**. Почему модуль **Backup and Migrate** попал в меню **Структура**, не очень понятно, но объяснимо.

Идея модуля следующая: мы задаем места, откуда берем данные, то есть базу данных и директорию с файлами. По умолчанию файлы располагаются в sites/default/files — это изменяемые файлы. Но, в частности, это означает, что файлы нашей темы нужно бэкапить отдельно. Файлы самого Drupal тоже лучше забэкапить для быстроты восстановления.

Мы можем делать бэкап вручную, а можем задать его расписание. Самое простое — сделать быстрый бэкап.

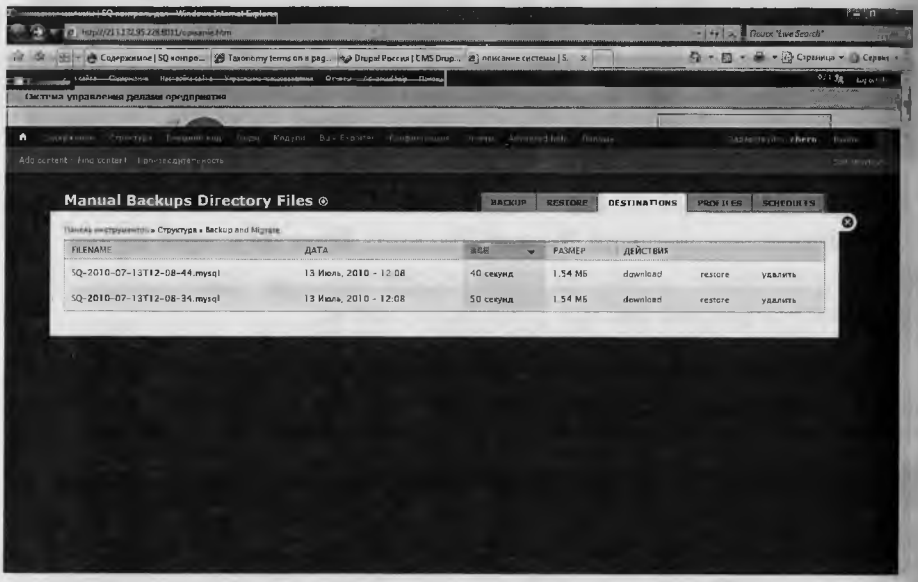
Заметьте: мы выбрали **Manual Backups Directory**. При щелчке по кнопке модуль сам создаст нужные директории и выполнит бэкап. Нужно сделать бэкап как файлов, так и базы данных.

После этого можно зайти в пункт **Destinations** и посмотреть список файлов для **Manual Backups Directory**.



Здесь мы видим, куда будут складываться файлы при ручном бэкапе и при бэкапе по расписанию. Активировав модуль для файлов, мы автоматически создадим еще одну директорию, в которую будут складываться бэкапы файлов. Сразу возникает вопрос: а как получается забэкапить файлы из директории sites/default/files, чтобы при этом они помещались внутрь нее же? Оказывается, в настройках бэкапа можно указать список игнорируемых директорий.

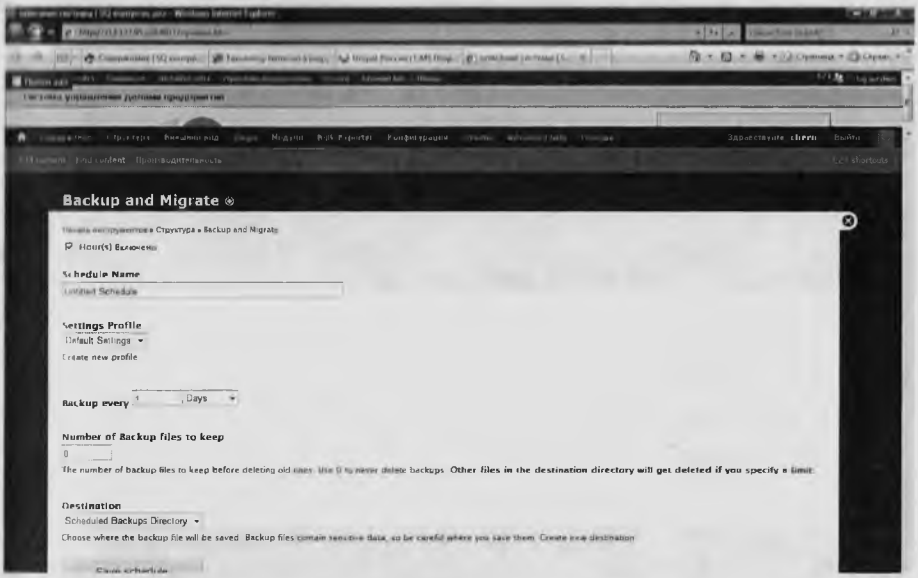
Список всех выполненных бэкапов можно посмотреть, перейдя по линку **list files** для соответствующей директории. Итак, переходим в **list files** для **Manual Backups Directory**.



Отсюда потом мы можем восстановить файлы. Однако делать ручной бэкап достаточно трудоемко. Поэтому лучше настроить расписание и делать бэкапы автоматически. Это делается через меню **Shedules**. Переходим в него и щелкаем по кнопке **Add Schedule**.

Полезная опция в данном меню — **Number of Backup files to keep**. Она определяет число файлов, которое должно храниться в директории, прежде чем начать стирать их по кругу.

Для тестового сервера можно далее не заморачиваться. Для рабочего же необходимо создать новый профиль, отличный от стандартного, выполнив команду **Create new profile** ⇒ **Add Профиль**.



Нас интересуют **Advanced Options**.

Для рабочего сервера надо отправлять письмо, если бэкап не прошел, и, возможно, переводить сайт в автономный режим на момент бэкапа. Рекомендуется хотя бы раз в неделю переносить директорию с бэкапами на другой носитель — не на том же сервере.

ПРИМЕЧАНИЕ

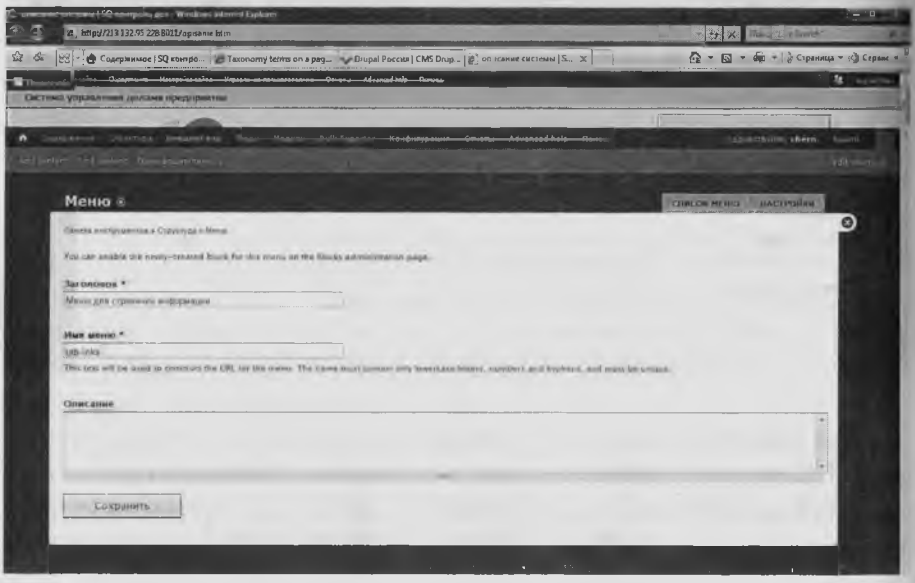
Проведите как минимум одну тренировку по восстановлению всей конфигурации вашего сайта и запишите процедуру восстановления. Можно сделать это на другом сервере. Мы собираемся сделать это, когда будем перемещать систему с тестового сервера на основной.

Закладки на странице

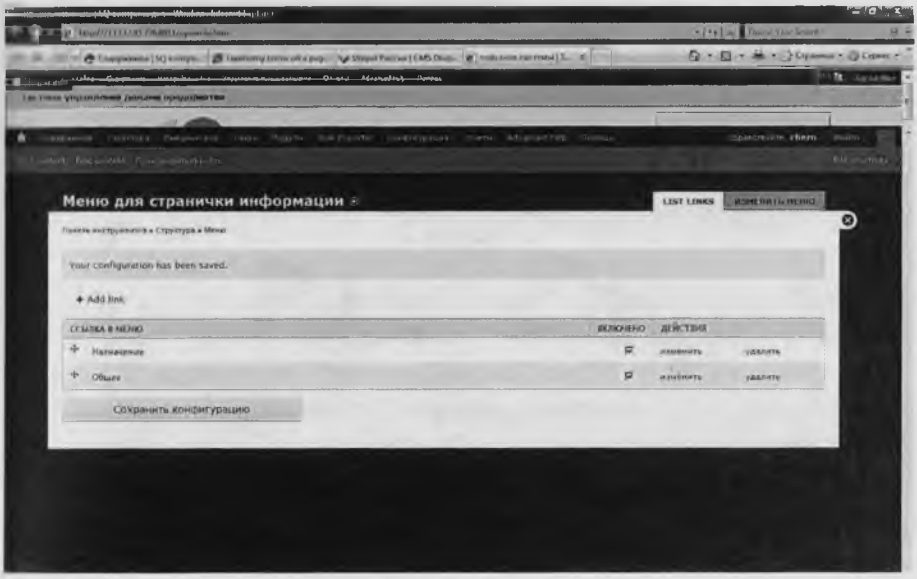
- Общая идея
- Новое меню
- Создание шаблона
- Результат

Мы хотели еще создать закладки на странице. Нам нужно открыть страницу и показать на ней несколько закладок, при щелчке по которым будет открываться соответствующий пункт. Есть различные модули закладок, но мы вполне можем обойтись тем, что уже знаем.

Мы заведем для таких страниц новый тип материала, а в node--type.tpl.php для этого типа поместим блок, который покажет закладки. Сами закладки будут формироваться на основе меню. Давайте заведем новое меню (при вводе надо указать на то, что мы хотим **Редактировать URL path**).



После этого введем нужные пункты.



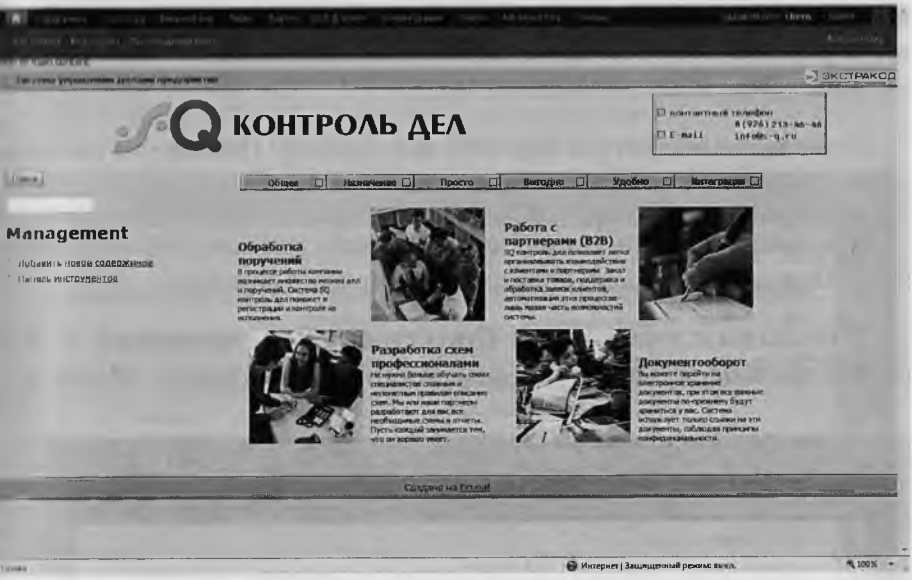
Отметим, что мы должны сначала создать страницы с информацией, а потом уже формировать пункты меню, которые на них указывают.

Теперь добавим показ меню в файл шаблона. Текст получается простой и незамысловатый:

```
<?php
// $Id$
?>

<table border=0 cellspacing=0 cellpadding=0
height=16 align=center>
 <tr>
   <td width=100% valign="top" height=20>
     <table border=0 cellspacing=0 cellpadding=0
align=center>
       <tr>
         <?php
```

```
            $menu = menu_tree_all_data('menu-tab-
links');
            $imgpath = base_path() . path_to_theme()
. '/images';
            foreach($menu as $link) {
              print '<td width=122 height=22
background="' . $imgpath . '/head_scr_3.gif"
style="cursor:pointer;" ';
              print 'title="' . $link['link']
['localized_options']['attributes']['title'] . '"
onClick=\'location.href="' . $link['link']['href'] .
'"\' onMouseOut=\'this.background="' . $imgpath . '/
head_scr_3.gif"\' onMouseOver=\'this.background="' .
$imgpath . '/head_scr_4.gif"\'>';
              print '<div align="center"><b><font
style="font-size: 9pt" face="Tahoma">' . $link['link']
['title'] . '</font></b></div></td>';
            }
          ?>
        </tr>
      </table>
    </td>
  </tr>
</table>
<br>
<table width=750 border=0 id=tab class=bodymark0
cellspacing=0 cellpadding=0 align=center>
  <tr>
    <td align=left>
      <?php print $content; ?>
    </td>
  </tr>
</table>
```

Оставим здесь свою разметку для примера. Вы, конечно, должны оформить меню закладок по-своему. А результат работы выглядит так:



ПРИМЕЧАНИЕ

Небольшая проблема возникла с тем, что значение `$link['link']['href']` приходит, например, как **node/46**. Поэтому иногда приходится добавлять символ `/` к началу этой ссылки при формировании ее из PHP.

Новости

- Новый тип **Новость**
- Модуль **Date**
- Поле типа **Date**
- **View** для показа новостей
- **Анонс** (Teaser)
- Ввод анонсов
- Показ View в блоке

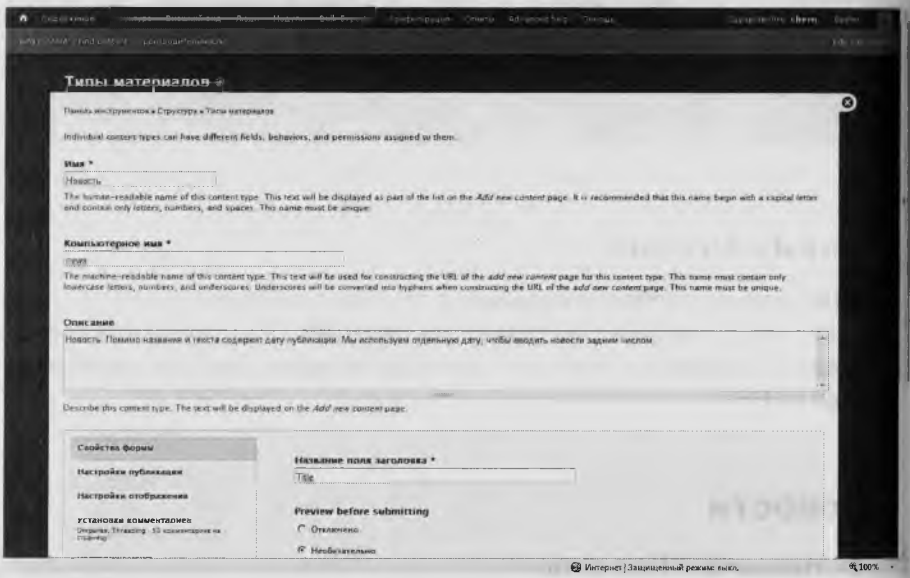
Что мы хотим от новостей? Для начала просто создадим окошко со списком всех новостей и кнопку на главной странице для перехода на

него. Туда мы введем все старые новости, а потом будем добавлять новые и развивать систему новостей: вводить экранчик с последними новостями, реализовывать листание и т. д.

Отдельного модуля для новостей нет. Почему? Потому что уже есть все необходимое для создания новостей.

Мы просто создадим новый тип материала для новостей и сформируем **View** для просмотра. Разберем этот процесс детально.

Сначала выполним команду **Структура ⇒ Типы материалов ⇒ Добавить тип содержимого** и создадим новый тип материала — **Новость**.



Все опции добавляем как обычно — по минимуму.

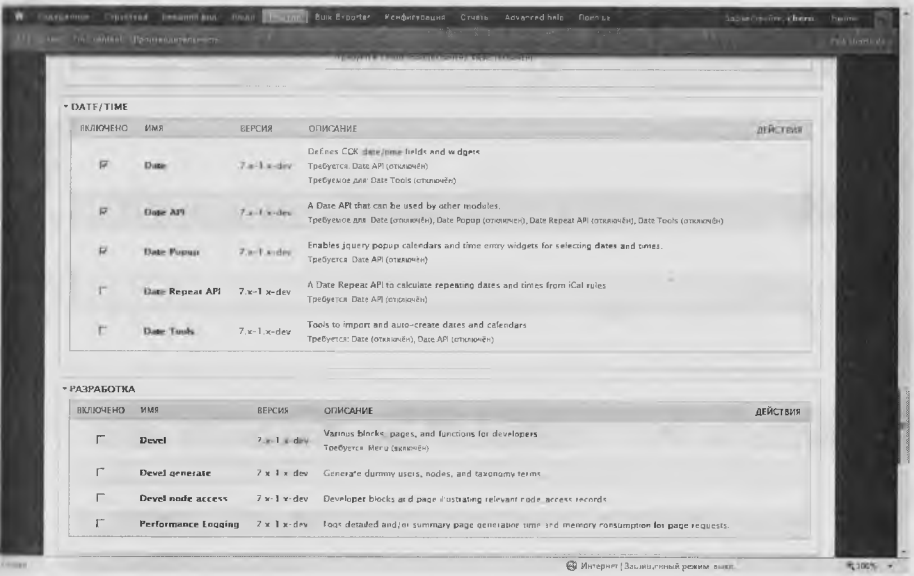
Добавляем поле **Дата публикации**. Оно должно быть датой, но у нас пока нельзя делать поля типа дат. Как обычно, для этого нужен дополнительный модуль.

Date (http://drupal.org/project/date) — модуль для работы с датами и создания полей типа дат в CCK. Если у вас PHP версии выше 5.3, то может понадобиться патч.

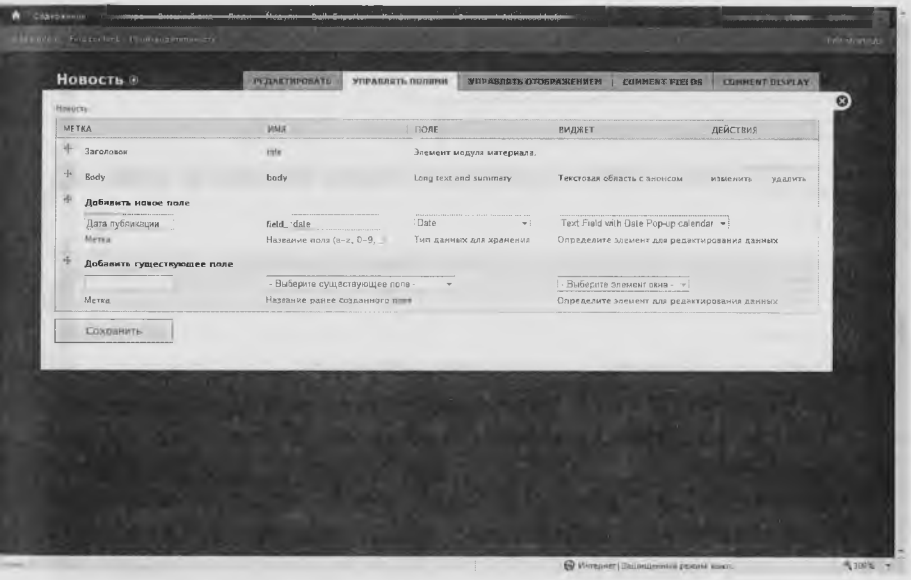
Ставим патч http://drupal.org/files/issues/date-6.x-2.x-549884-30.patch. По идее в Drupal 7 он не нужен.

Date состоит из нескольких модулей и, скопировав его, нужно будет выбрать набор, который мы хотим использовать.

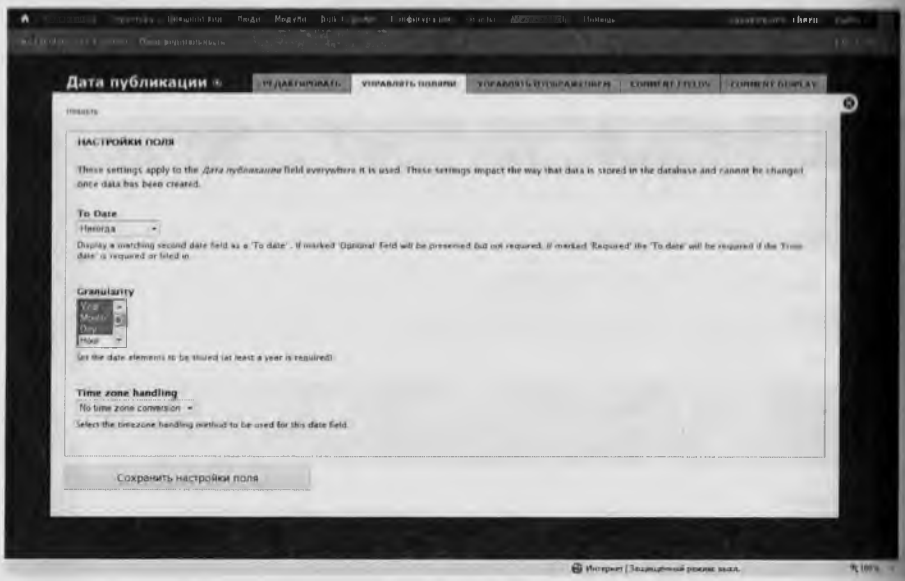
Сначала мы ограничились следующим набором модулей.



После включения мы можем создать поле типа **Дата** (Date).

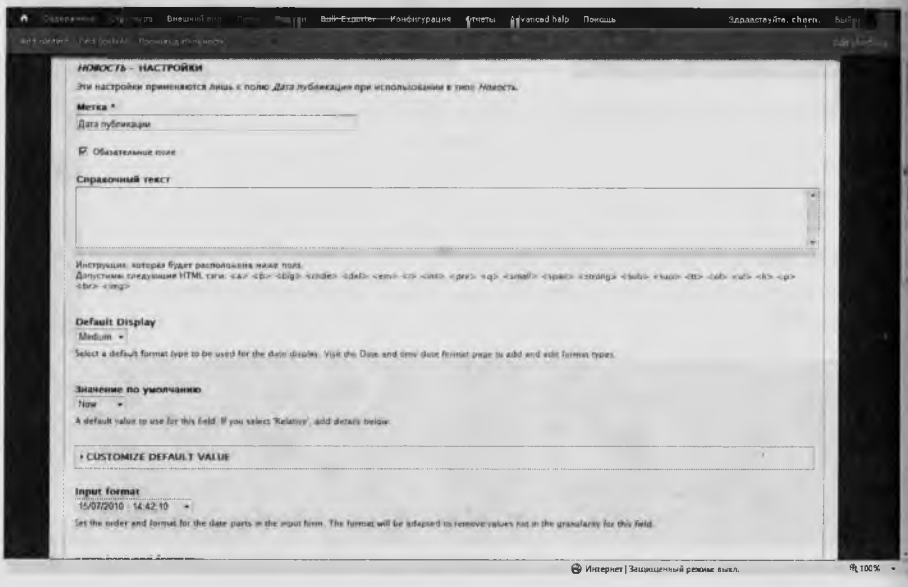


В дополнительных опциях укажем:



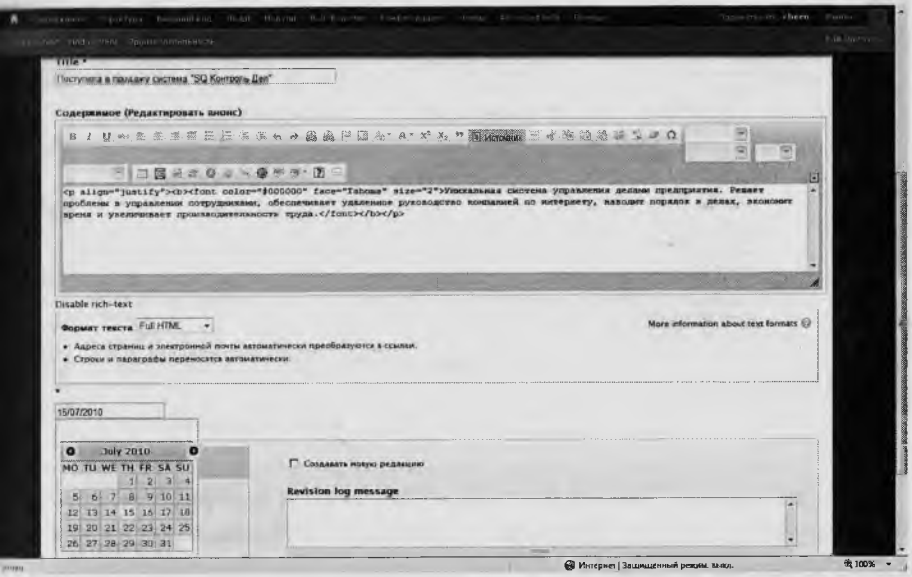
Необходимо вводить даты с точностью до дня. При этом нам не нужно преобразование **Time zone**.

Опции вводятся в двух экранах: первый (выше) содержит общие настройки для поля `field_date`; во втором опции настраиваются при использовании поля `field_date` в типе **Новость**.



Мы укажем на обязательность даты и зададим формат ввода dd/mm/yyyy. На рисунке показана дата работы с данным разделом — 15/07/2010.

Начинаем вводить новости.



В поле отображается календарь для ввода даты. Можно использовать его, а можно вводить данные вручную.

Вводим дату новости. Наша новость была опубликована аж в 2001 году. Такую дату проще ввести вручную.

Введем несколько новостей, благо они у нас присутствовали на старом сайте. В новости можно включать произвольный HTML-текст, в том числе и картинки.

Введем сразу все старые новости, чтобы можно было получить нужную нам страницу целиком.

Для просмотра новостей нам понадобится **View**. Мы уже создавали **View** для отзывов, аналогично сделаем **View** и для новостей. Но новости мы оформим в специальном стиле (как оформляли отзывы).

Отметим, что список новостей будет общим для всех русскоязычных сайтов. Мы будем изменять только оформление при выводе новостей на разные сайты.

Итак, создаем **View**.

Что мы ввели? Дату публикации, название и содержимое новости. Фильтруем мы по типу **Новость** и признаку **Опубликовано**, а также по дате в обратном порядке.

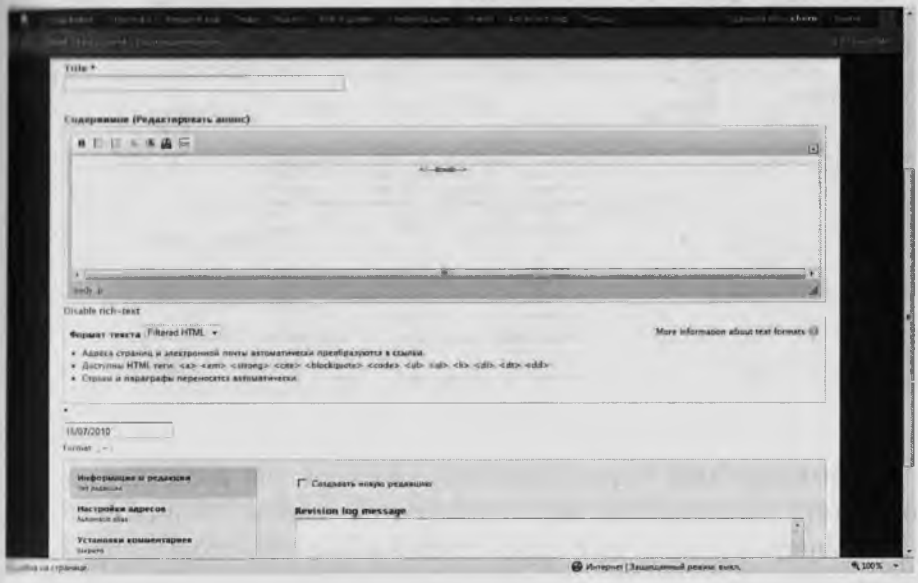
Кроме того, мы создаем дисплей типа **Страница** с путем news.htm. На странице указывается вывод новостей без ограничения. После этого уже можно использовать данную страницу для показа. Осталось только отформатировать, но вы уже знаете, как это делается. В нашем случае надо создать два файла шаблона с названиями views-view--sq-task-control-news.tpl.php и views-view-fields--sq-task-control-news.tpl.php.

Еще добавим блок с последними новостями для показа в произвольном месте. Но для этого нужно научиться работать с анонсами (teaser).

Для материала может быть задан текст анонса. Он обычно используется при анонсировании материалов на первой (и на любой другой) странице внутри некоторого блока.

Размер текста анонса задается в меню **Настройки отображения** для типа материала. Признак называется **Length of trimmed content** (Длина усеченных материалов). Однако рекомендуется управлять анонсами вручную, поскольку автоматическая обрезка работает не очень правильно.

Чтобы определить, где у нас анонс, а где материал, при вводе содержания нужно использовать разделитель <!--break-->. Он отделяет начальную часть материала, которая будет использоваться как анонс. В редакторе есть специальная кнопка для ввода данного разделителя — крайняя справа. Щелкнем по ней.



Разделитель появился...

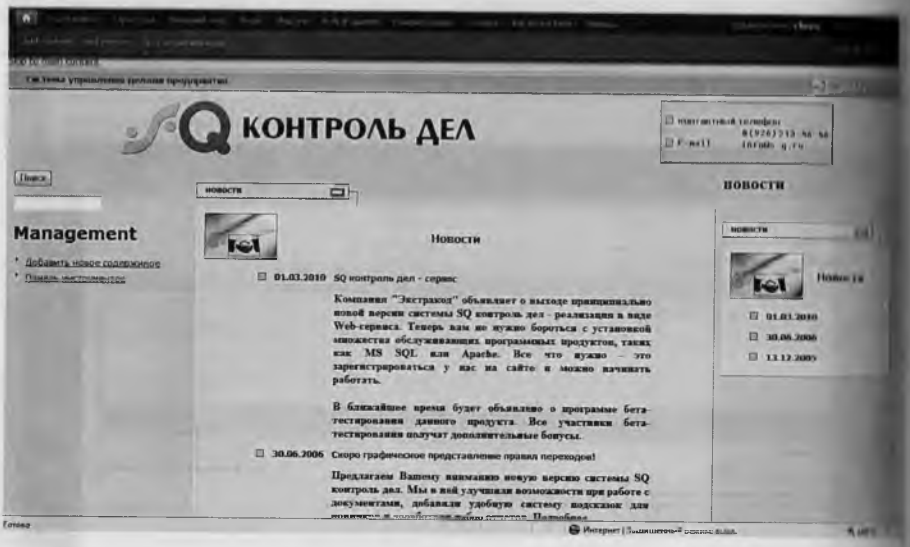
Иногда хочется создать анонс, который будет отличаться от начальных строк материала. В этом случае используется кнопка **Редактировать анонс**. При щелчке по ней анонс и материал будут независимы, и появится отдельное поле для ввода анонса.

Итак, мы создадим анонсы для последних материалов путем вставки разделителя и модифицируем **View**. Добавим в него на странице **Defaults** еще одно поле **Материал: Teaser**.

Далее активируем дополнительный дисплей вида «блок» и укажем, что не показываем на странице поле **Teaser**, а в блоке зададим отсутствие названия и содержания.

Однако нам все равно придется создать два специальных файла для форматирования всего блока и каждой его строки: views-view--sq-task-control-news--block.tpl.php и views-view-fields--sq-task-control-news--block.tpl.php.

Результат выглядит примерно следующим образом.



Страница была отформатирована, но блок мы пока поленились оформить должным образом, поэтому он выводится с форматированием страницы. Думаем, что вы уже способны доделать оформление блока в нужном виде.

Работа с формами

- API форм
- Три функции форм
- Пример формы
- Обработка формы
- Webform

Наиболее гибким способом создания форм в Drupal является использование специального API для формирования, проверки и обработки данных формы. Идея API в том, что форма описывается в виде массива PHP в конструкторе форм. Этот конструктор используется дважды: при создании и показе формы, а также при проверке и обработке данных. По сути, чтобы описать форму, мы должны создать три

функции (идентификатор следует заменить уникальным названием формы):

- `function идентификатор($form_state)` — конструктор формы, должен вернуть массив с описанием формы.

Далее этот массив формы можно вывести на страницу при помощи оператора `print drupal_get_form('идентификатор')`. Оператор можно использовать как в файле шаблона tpl, так и в материале, только нужно включить обработку кода PHP;

- `function идентификатор_validate($form, &$form_state)` — проверка формы. Может выдавать ошибку при помощи `form_set_error`;

- `function идентификатор_submit($form, &$form_state)` — запись полученных данных. Может выдавать стандартное сообщение при помощи функции `drupal_set_message`, а вызывать `drupal_goto` для перехода на другую страницу.

Массив представляет собой список полей для ввода и управляющих элементов, описанный в http://api.drupal.org/api/drupal/developer--topics--forms_api_reference.html/7. Это очень похоже на описание элементов HTML, только со своей спецификой. Также отметим, что некоторые поля могут вкладываться друг в друга, например в **radios** можно вложить несколько **radio**, в **checkboxes** — несколько **checkbox**, а в **fieldset** — любое количество других элементов. По сути, все формы в интерфейсе настройки — это и есть такие **fieldset** с набором полей, а их внешний вид — стандартное их представление в теме **Garland**. Также существует возможность создавать произвольный **markup** внутри формы. Мы используем это при форматировании своей формы.

Для примера создадим простую форму и выполним ее обработку. Форма будет регистрировать новых пользователей в простейшем виде и записывать их в базу данных. Запись мы опустим, сосредоточимся только на создании и обработке данных формы. Подобная форма использовалась в нашей партнерской программе, и мы пока не будем менять способ авторизации в этой части сайта для совместимости с прошлым вариантом. Мы возьмем эту форму из примера с сайта drupal.org.

Сначала надо сформировать функцию формы. Ее мы размещаем в файле template.php для нашей темы:

```
function sq_test_form($form_state) {
  $form['description'] = array(
    '#type' => 'item',
    '#title' => t('A form with a submit handler'),
  );
  $form['name'] = array(
    '#type' => 'fieldset',
    '#title' => t('Name'),
    '#collapsible' => TRUE,
    '#collapsed' => FALSE,
  );
  $form['name']['first'] = array(
    '#type' => 'textfield',
    '#title' => t('First name'),
    '#required' => TRUE,
    '#default_value' => "First name",
    '#description' => "Please enter your first
name.",
    '#size' => 20,
    '#maxlength' => 20,
  );
  $form['name']['last'] = array(
    '#type' => 'textfield',
    '#title' => t('Last name'),
    '#required' => TRUE,
  );
  $form['year_of_birth'] = array(
    '#type' => 'textfield',
    '#title' => "Year of birth",
    '#description' => 'Format is "YYYY"',
  );
  $form['submit'] = array(
    '#type' => 'submit',
    '#value' => 'Submit',
```

```
  );
  return $form;
}
```

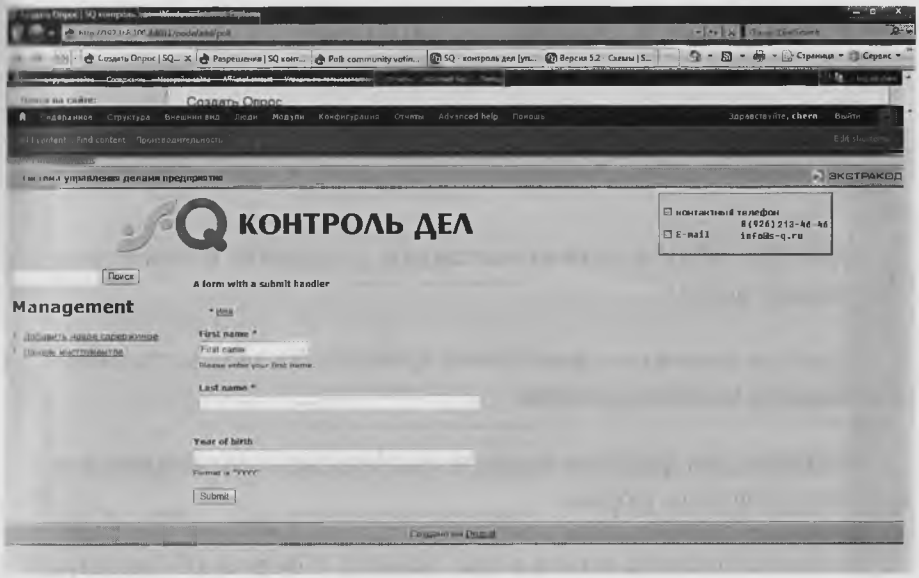
Форма носит название `sq_test_form`, по этому имени мы к ней и будем обращаться.

Там, где нужно показать форму, мы вставляем:

```
<?php print drupal_render(drupal_get_form('sq_test_
form')); ?>
```

Такая схема сработала в наших файлах шаблонов, но почему-то не сработала внутри материала. Скорее всего, это временный баг.

Тем не менее смотрим, что у нас получилось.



Как было указано выше, создаем функции для обработки:

```
function sq_test_form_submit($form, &$form_state) {

  drupal_set_message(t('The form has been
submitted. name="@first @last", year of birth=@year_
of_birth',
```

```
        array('@first' => $form_state['values']
['first'], '@last' => $form_state['values']['last'],
'@year_of_birth' => $form_state['values']['year_of_
birth'])));

    $form_state['redirect'] = '/node/1';

    drupal_redirect_form($form_state);

  }

  function sq_test_form_validate($form, &$form_state)
{

    $year_of_birth = $form_state['values']['year_of_
birth'];

    if ($year_of_birth && ($year_of_birth < 1900 ||
$year_of_birth > 2000)) {

      form_set_error('year_of_birth', 'Enter a year
between 1900 and 2000.');

    }

  }
```

Функция сабмита должна выставлять сообщение и переводить нас на страницу node/1.

Функция проверки осуществляет проверку правильности ввода года и сообщает о наличии проблем.

Включите эти функции также в template.php и посмотрите, как будет вести себя наша форма.

Осталось добавить запись в базу данных, и форма для регистрации пользователей готова.

Довольно много примеров форм находится в http://api.drupal.org/api/drupal/developer--examples--form_example--form_example_tutorial.inc/7.

Самое простое — создавать формы, используя уже существующие примеры.

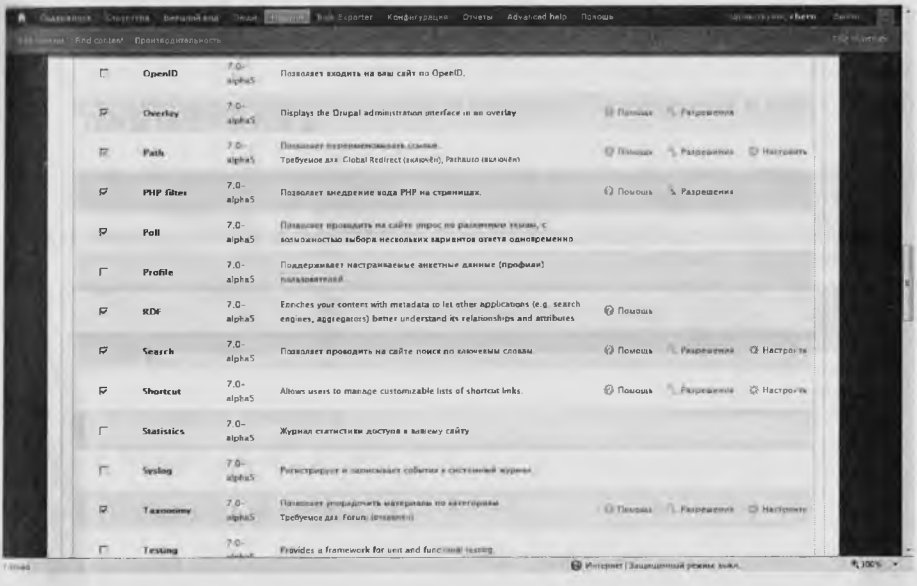
ПРИМЕЧАНИЕ

Помимо использования API существует также ряд модулей для простого создания и использования форм. Например, при помощи модуля **Webform** (http://drupal.org/project/webform) можно достаточно легко создать материал с формой. Данные формы сохраняются в отдельную таблицу, которую можно просмотреть и использовать для последующей обработки.

Опросы

- Модуль Poll
- Права на опросы
- Создание опроса
- Шаблоны для настройки внешнего вида

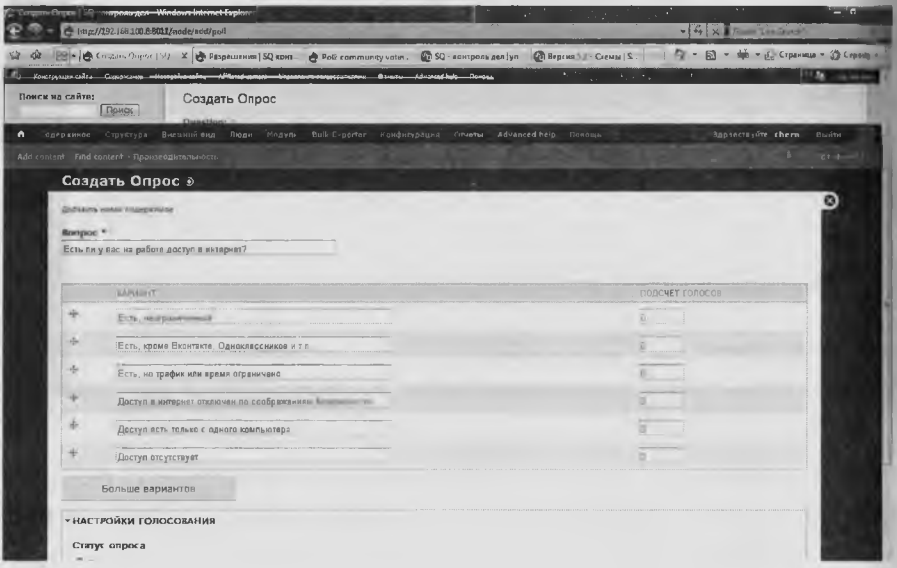
Опросы — это сейчас настолько стандартная возможность, что в Drupal 7 модуль проведения опросов входит в ядро. Это модуль **Poll**, и наконец, пришло время его включить.



Нам нужно дать право голосовать, а также просматривать результаты голосования всем пользователям. Ставим флажки для **Vote on polls** и **View voting results**.

Итак, что же изменилось? Обычно это довольно сложно определить, не изучив README.txt о дополнительных модулях — там приводится описание каждого модуля со списком привнесенных возможностей. Кстати, данный файл доступен через функцию **Advanced Help**.

Прежде всего появился новый тип материала – **Опрос**. Надо придумать какой-нибудь простенький опрос, например: «Есть ли у вас на работе доступ в Интернет»? Заводим вопрос и разные варианты ответов.



Появляется блок с названием **Последний опрос**, в котором отображается последнее голосование. Мы располагаем его в нужном блоке и указываем страницы, на которых хотим его показывать (или не показывать). Все эти функции мы уже разбирали.

Кроме того, необходимо настроить HTML-разметку блока, то есть его оформление. Здесь нас выручает любимый модуль **Devel**.

Выясняем, что по умолчанию оформление управляется шаблоном poll-vote.tpl.php, но для блока его можно переопределить шаблоном poll-vote--block.tpl.php. Снаружи все преобразования «одеваются» в block.tpl.php, который может переопределяться как шаблоном block-poll.tpl.php, так и block-poll-0.tpl.php. Вопрос в том, насколько можно разделить оформление блока и оформление самого голосования.

Существует множество шаблонов для стандартного модуля голосования: для просмотра результатов, для рисования прямоугольника и т. п. (подробное описание см. в приложении Шаблоны-кандидаты). В общем, все как всегда. Лучше день потерять, потом за пять минут долететь. Но не думаю, что нам понадобится много различных шаблонов для оформления голосования. Иначе все будет слишком пестро. В крайнем случае, управляйте признаками таксономии для изменения внешнего вида внутри PHP.

ПРИМЕЧАНИЕ

Есть также довольно много модулей, которые позволяют оценить материалы. Познакомиться с ними можно по адресу: http://drupal.org/node/211269.

Книги

- Что такое книжка
- Создание книги
- Добавление страниц
- Показ оглавления
- Версия для печати
- Переупорядочивание глав
- BookMadeSimple

Отдельная страница для каждого материала — это, конечно, хорошо. Но в какой-то момент возникает необходимость публиковать более сложные материалы с иерархическим оглавлением и т. д. Проще говоря, книги.

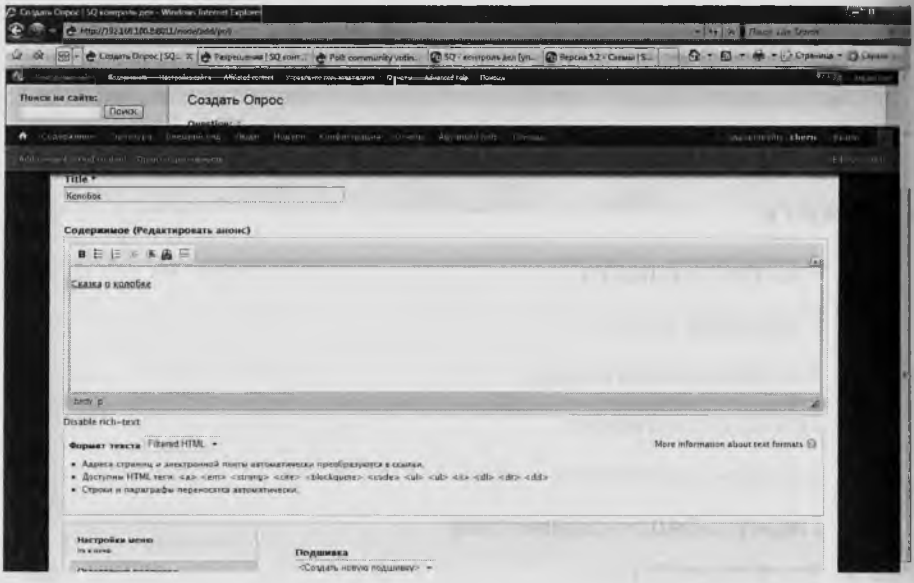
Что такое книга? Это набор материалов, которые собраны в некую иерархическую структуру — оглавление. Мы можем создавать и перемещать материалы в оглавлении.

При создании книги возникает отдельный блок — оглавление, которое размещается на странице (обычно в левой колонке) и отображается только на страницах, принадлежащих данной книге. Это теория. Теперь перейдем к практике.

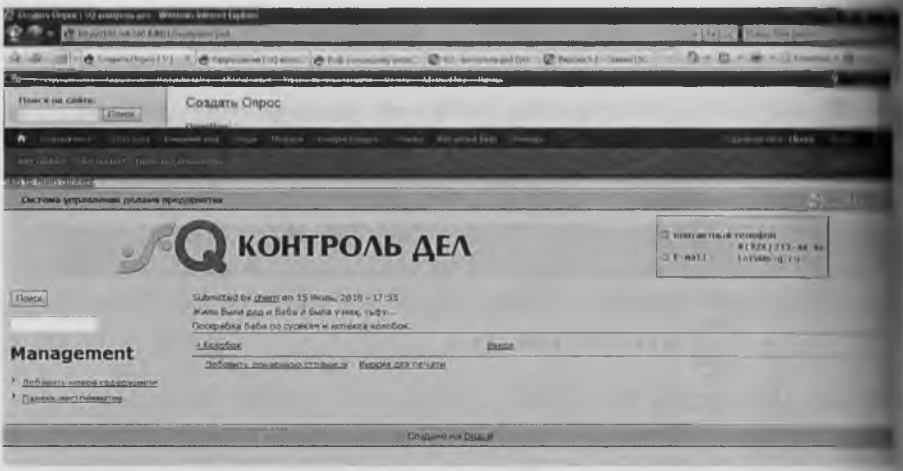
Модуль **Book** является стандартным и входит в дополнительные модули ядра. Включим его и попробуем создать простую книгу, например «Колобок».

Модуль включил для нас новый тип страницы под названием **Book page**. Этот тип используется для оформления страниц. Вы можете дополнять его, а также вставлять в книгу страницы произвольных типов. Заметим, что на данный тип, как и на все остальные созданные типы, надо давать права вручную.

Для начала создадим титульную страницу книги.

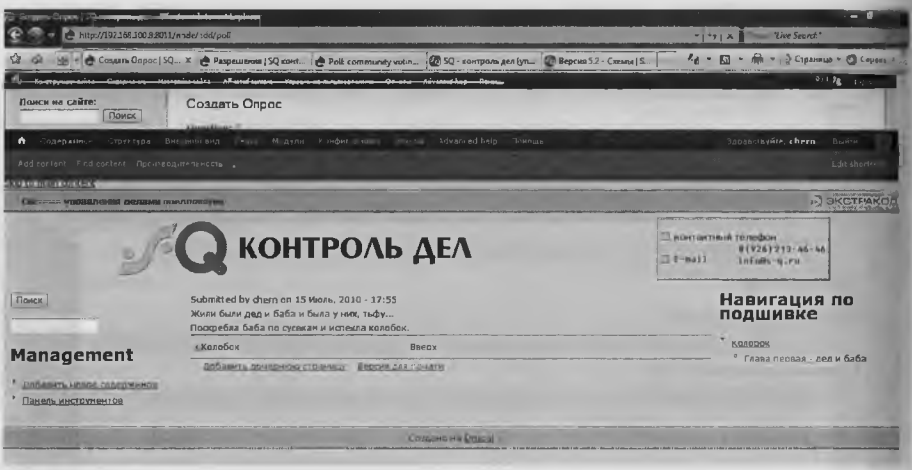


Получаем следующий результат.



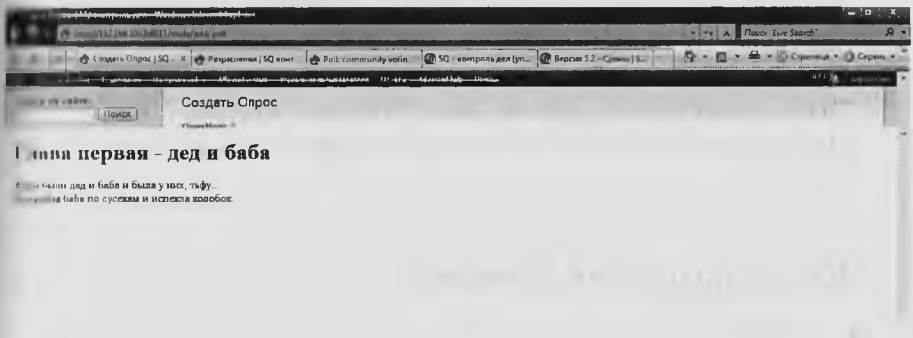
Что мы видим? Базовое оформление, которое надо подстраивать стилями или настройками темы. Это вы знаете. Далее — оглавления пока тоже нет. Оглавление книги — это отдельный блок, который нужно разместить на странице. Переходим к блокам и размещаем оглавление в правой колонке. Блок называется **Навигация по подшивке**.

Смотрим, что получилось.

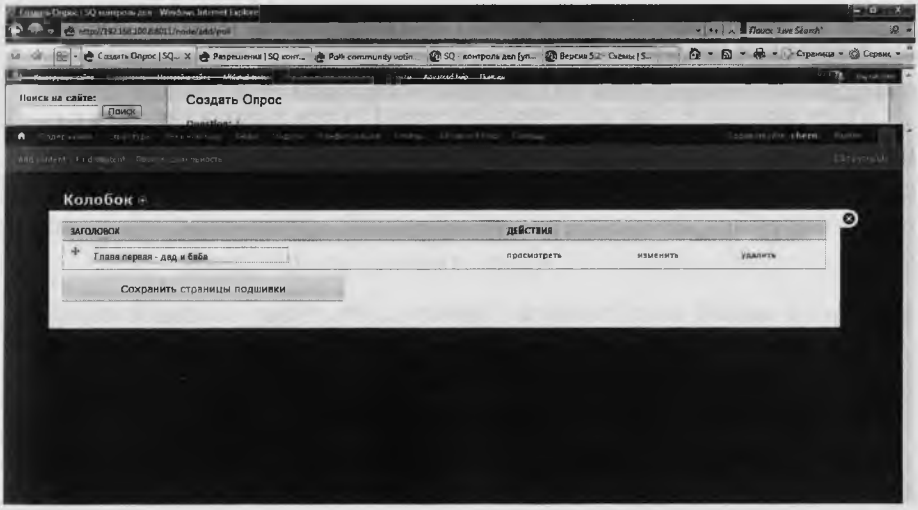


Мы можем перемещаться как в левом меню, так и при помощи ссылок в нижней части раздела, а также ссылок на нижележащие главы.

Есть еще дополнительная кнопка — **Версия для печати**. Она формирует максимально простое оформление.



Чтобы добавить страницу на первый уровень, нужно встать на титульную страницу книги и указать, что мы хотим добавить страницу. Если же страницу надо добавить внутрь главы, достаточно перейти на главу. В целом, все просто. Иногда приходится переупорядочивать строки через меню **Содержимое** ⇒ **Подшивки**. Мы выбираем одну из книг и отмечаем опцию **Изменить порядок и наименования**. А затем можем переставлять пункты при помощи перетаскивания и менять названия глав прямо отсюда.



Отметим, что мы пока отключали возможность ввода комментариев к страницам книги.

ПРИМЕЧАНИЕ

Есть еще модуль **BookMadeSimple** (http://drupal.org/project/BookMadeSimple), который позволяет при вставке нового материала прямо из книги выбрать его тип. В Drupal 6 с ним почему-то не сложились отношения. После его установки исчезли все ссылки в основном навигационном меню. Что будет в Drupal 7, неизвестно. Версии модуля еще нет. Попробуйте, может быть, у вас уже все получится.

Обзор модулей поиска

- Search
- Google
- Apache Solr, Lucene

Поиск по сайту — это отдельная и глобальная задача, которая, как и другие задачи, может решаться в Drupal различными способами. Здесь мы разберем основные способы организации поиска на сайте и сравним их.

Базовый модуль Search

Идея модуля в том, что периодически (путем вызова индекса `cron`) производится индексация всех материалов, в результате чего появляется возможность поиска на основе данного индекса. Эти модули легко инсталлируются, но имеют ряд ограничений, которые частично снимаются другими модулями. Разберем имеющиеся проблемы.

1. Поиск может выполняться только по полному совпадению слова в тексте. Это решается при помощи модулей для морфологического поиска (stemming). Для английского языка модуль называется **Porter-stemmer**. Есть модуль для подключения русского языка — **Russian Stemmer**. Эти модули пытаются выделить корень из каждого слова, и индекс строится уже по корням.
2. Модуль **Search** индексирует именно материалы, а не страницы. А материалы могут показываться самыми различными способами. Данная проблема решается инсталляцией модуля **Search by Page**,

который имеет очень широкий спектр настроек. В частности, он поддерживает:

- индексацию материала в том виде, как он показан темой;
- выбор типов материалов для индексации;
- индексацию View и других страниц не являющихся материалами;
- возможность показа страниц только на нужном языке;
- возможность иметь несколько блоков поиска и страниц результатов для поиска в разных частях сайта или мультисайтовости.

1. Следующие проблемы — это отсутствие масштабируемости. Модули не могут корректно работать с объемными сайтами, поскольку модуль **Node** основан на SQL-методах. Эта проблема, к сожалению, никак не решается, и базовый метод поиска применим только к небольшим сайтам. Но «небольшой» в данном случае — это более тысячи страниц.

Итак, вывод — базовый модуль можно использовать для небольших сайтов, но необходимо установить ряд дополнительных модулей. Можно также инсталлировать модуль **Search Files** для поиска внутри файлов.

Модули для подключения поисковиков (Google)

Следующий класс модулей — модули для интеграции с поисковыми системами. Смысл их в том, что страницы индексируются при помощи поисковой системы, например Google, после чего поиск организуется путем запроса к нему. Модуль **Google Custom Search Engine** обеспечивает взаимодействие между Google CSE и Drupal Search API. То есть, в принципе, можно пользоваться поиском Google, не подключая его к Drupal. Модуль просто обеспечивает дополнительную функциональность при поиске.

Данное решение дает весьма мощные возможности поиска, но также имеет ряд ограничений.

1. Индексируются целые страницы. В результате, вводя для поиска слово из базового меню, мы получим все страницы нашего сайта.
2. Время реиндексации неизвестно. Google сам определяет, когда индексировать страницы. В результате, если материалы активно меня-

ются, то велик шанс, что пользователь будет получать устаревшую информацию.

Использовать данный модуль очень просто, но мы теряем контроль над процессом индексации. Можно использовать его для индексации объемных сайтов, которые меняются относительно редко. Рекомендуется установить также модуль **XML sitemap** для предоставления информации поисковой системе.

Модули для интеграции с поисковыми движками (Solr, Lucene)

Данный тип модулей позволяет интегрировать Drupal с отдельным поисковым движком, установленным на вашем сервере. Наиболее часто используются движки Apache Solr и Lucene. Первый имеет большую функциональность, но написан на Java, в результате вам нужно иметь, как минимум, Apache Tomcat. Второй движок проще и написан на PHP, но также имеет множество полезных возможностей. Опишем вкратце их отличия от базового модуля поиска.

1. Возможность использования расширенного синтаксиса. Можно делать сложные запросы с условиями И/ИЛИ и т. д.
2. Сортировка результатов поиска.
3. Фасетный поиск, то есть поиск по множественным классификаторам.

Модуль **Lucence** интегрируется со стандартными модулями для морфологического поиска (Porter-stemmer). Модуль **Apache Solr** требует дополнительный модуль **Apache Solr Multilingual**.

Итог следующий: **Lucence** можно и рекомендуется ставить как замену стандартному методу поиска даже для небольших сайтов (понятно, что если у вас сайт из 10 страниц, то это вам не нужно, но если страниц больше 1000, то пора). **Apache Solr** предназначен для объемных сайтов и позволяет даже разделять поисковый сервер и сервер сайта. В результате, если вам нужна масштабируемость, выбирайте именно **Apache Solr**.

Таким образом, вы можете выбрать систему поиска исходя из собственных требований и того, насколько вы готовы возиться с ее настройкой.

Статистика и мониторинг

- Statistics и Statistics Advanced Settings
- Настройка
- Отчет о состоянии
- Доступные обновления
- Выполнение обновления
- Недавние записи в системном журнале
- Источники посетителей
- Ошибки «отказ в доступе»
- Ошибки «страница не найдена«
- Популярные поисковые запросы
- Последние посещения
- Список полей
- Популярные страницы
- Лучшие посетители
- Панель инструментов (Dashboard)

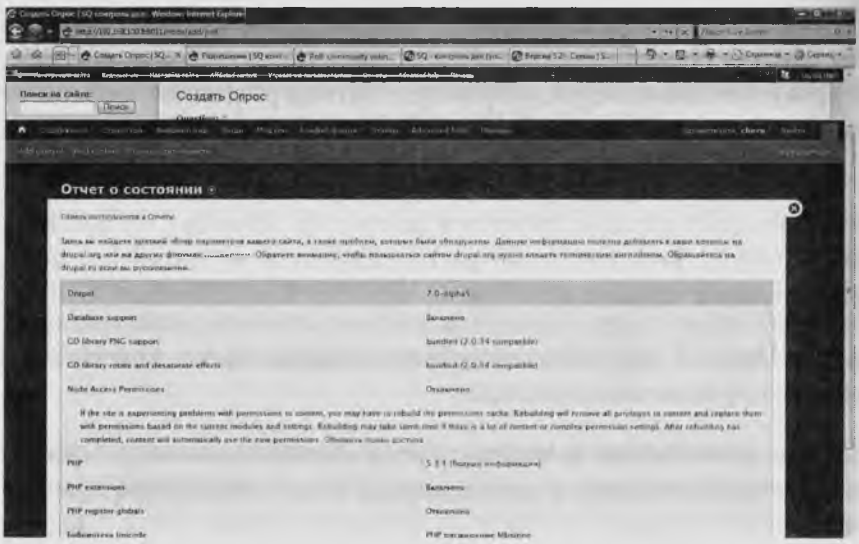
Нам необходимо следить за доступом к сайту. Информация о том, кто, когда, откуда и как часто смотрит наши страницы, поможет в будущем удобнее реорганизовать сайт и понять, какая информация больше интересует пользователей, а какая нет. За сбор различной информации отвечает базовый модуль ядра **Statistics**.

Помимо него сразу ставьте модуль **Statistics Advanced Settings** (http://drupal.org/project/statistics_advanced). Вообще, устанавливая некий модуль, неплохо посмотреть в алфавитном списке все модули с аналогичным названием.

Для Drupal 7 еще такого модуля нет, но надеюсь, что появится к моменту его официального выхода.

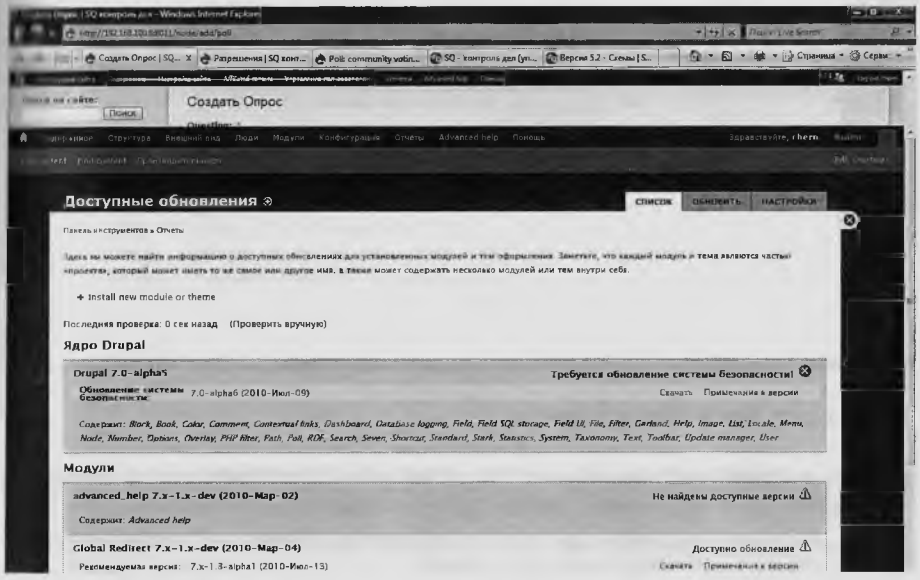
Итак, устанавливаем модули, в меню **Конфигурация ⇒ Статистика** включаем все опции и указываем срок, который будут храниться логи.

После установки **Statistics Advanced** появится еще одна закладка для настройки, но пока мы ее показать не можем. Отметим только, что для включения всех опций потребуется установить еще модуль **Browscap** (http://drupal.org/project/browscap). Он позволит регистрировать типы браузеров, которыми просматривают сайт. Мы установили некоторое количество модулей статистики и получили доступ к множеству отчетов. Давайте попробуем разобраться с появившимися возможностями. Первый, самый главный отчет — это, конечно, отчет о состоянии (**Отчеты ⇒ Отчет о состоянии**).



Его нужно проверять при установке новых модулей. Часто они могут что-либо потребовать.

При возникновении ошибок или предупреждений необходимо их устранять. Ну и, конечно, надо всегда обращать внимание на строки с информацией об обновлениях (**Отчеты ⇒ Доступные обновления**).

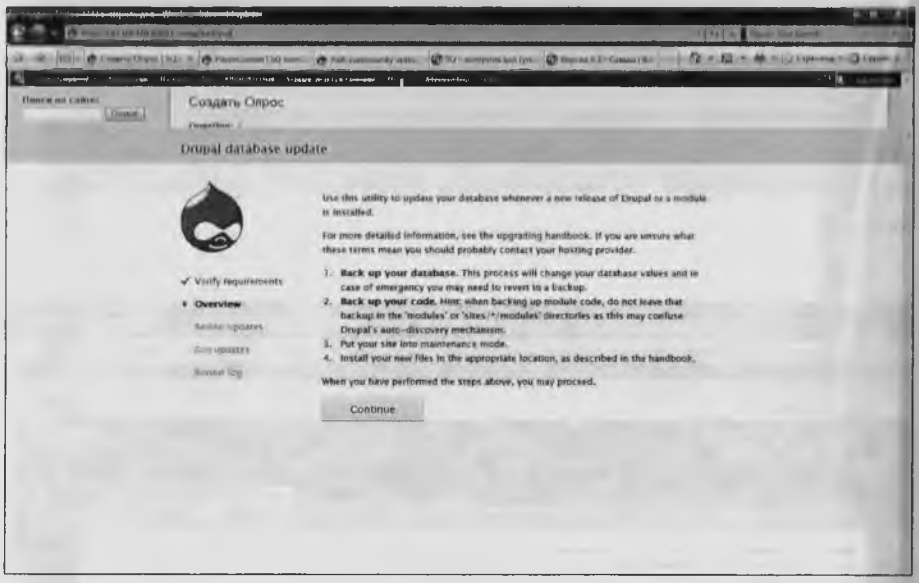


Обновления проверяются процедурой, запускаемой из `cron`, и при возникновении новых версий модулей их рекомендуется устанавливать.

С этим можно особо не спешить, но, если появляется предупреждение о том, что новая версия лечит прореху в безопасности, ее, конечно, следует поставить как можно быстрее. Подобные обновления будут выделены красным цветом.

В данном случае мы не спешим, поскольку все равно наша версия не рабочая, и особо важной информации на сервере нет.

Обычно для установки модуля нужно просто скопировать его поверх прошлого и выполнить файл update.php. Это делается от имени администратора — просто введите URL вашего сайта и update.php, например: http://www.example.com/update.php.



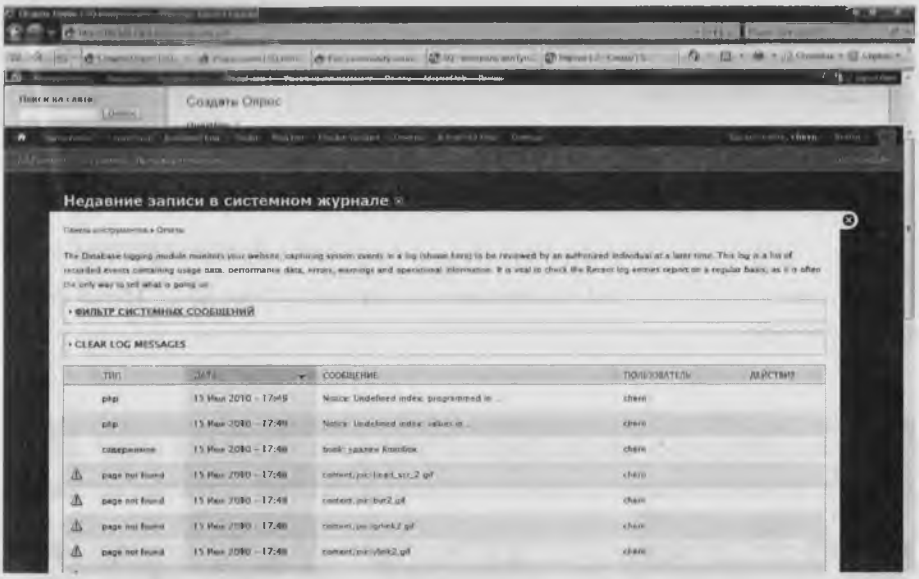
Здесь приведены операции, которые следует выполнить при апдейте:

1. Сделать бэкап базы данных.
2. Сделать бэкап кода. Подсказка: не оставляйте файлы с бэкапом в директориях с модулями, поскольку это может запутать Drupal.
3. Перевести сайт в режим поддержки.
4. Инсталлировать новые файлы в нужные места, то есть скопировать новые версии модулей поверх старых.

После этого можно продолжить процесс апдейта — проверить список апдейтов, запустить их на выполнение и посмотреть, что получилось.

Следующий отчет — системный журнал. В него записываются все странные события, которые произошли в системе. В принципе записываются и не странные, но нас интересуют именно проблемы. Вызвать журнал можно через опцию **Отчеты ⇒ Недавние записи в системном журнале**.

Мы видим, что в открывшемся окне присутствуют просто события, ошибки и предупреждения. Если сайт работает правильно, ни ошибок, ни предупреждений возникать не должно. Данный отчет надо просматривать раз в день, если ошибки были, и раз в неделю, если сайт достиг устойчивой работы.



Указав на текст сообщения, мы можем посмотреть детали, например:

Обычно предупреждения возникают при отсутствии информации по некоторому URL. Очень часто это проблемы с картинками.

В нашем случае проблемы появились, поскольку мы перенесли уже существующий сайт, и старые ссылки на картинки полностью не соответствуют новым.

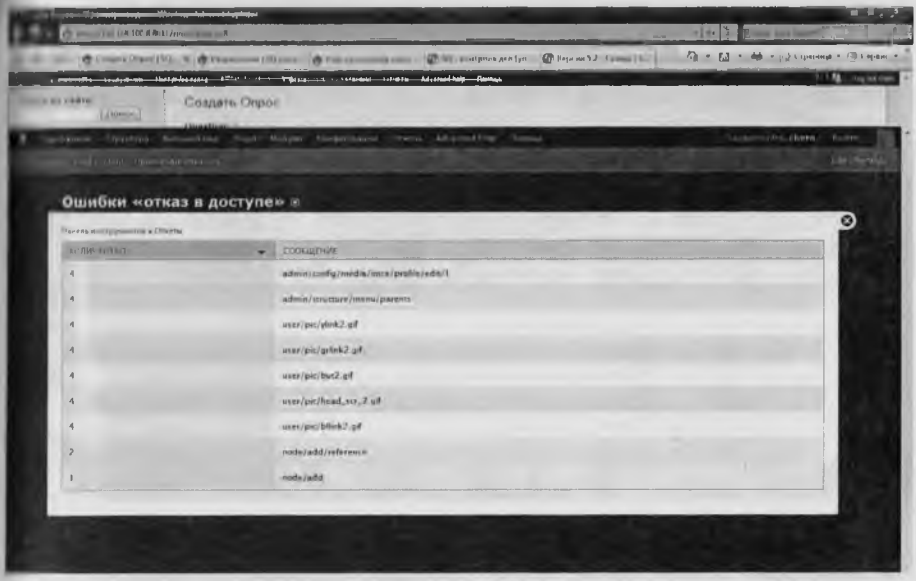
При создании сайта с нуля подобных ошибок должно быть меньше. В любом случае подобные ситуации надо анализировать и исправлять.

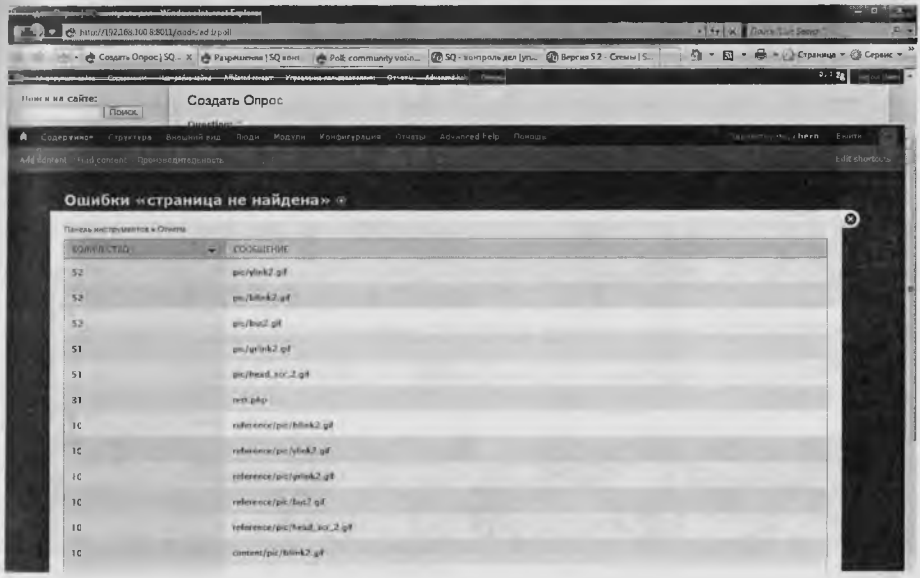
Далее следует отчет **Источники посетителей (Отчеты ⇒ Источники посетителей)**.

В отчете показываются сайты со ссылками на ваш сайт. При этом интересно, что регистрируются не только переходы на него, но и, например, показы баннеров, которые расположены у нас, но отображаются на других сайтах.

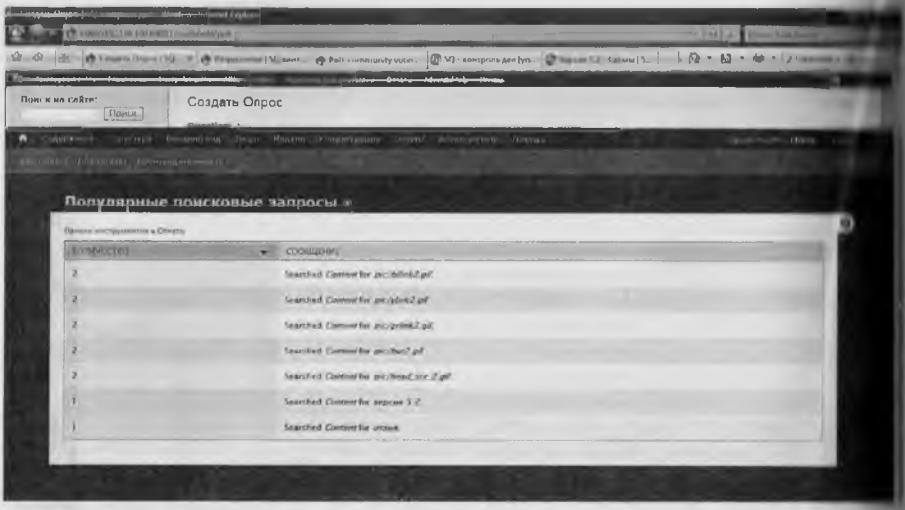
Тем не менее отчет покажет, насколько активно вы взаимодействуете с другими сайтами.

Далее два отчета, которые наиболее удобны для анализа проблем: **Ошибки «отказ в доступе»** и **Ошибки «страница не найдена»**. Они находятся все в том же меню **Отчеты**.



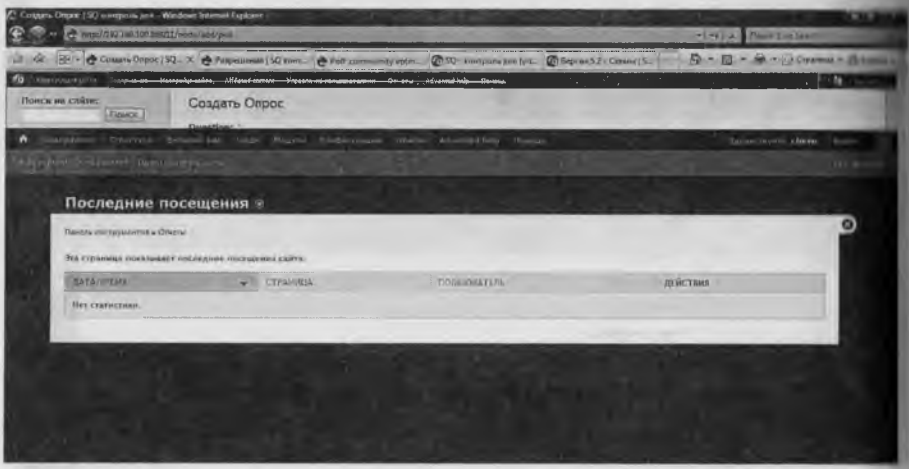


Понятно, что все ошибки надо разбирать и исправлять. В нашем случае иногда их невозможно исправить, поскольку отображаются ссылки с других сайтов на устаревшие ресурсы. Короче, во всем надо разбираться. Отчет **Популярные поисковые запросы** позволяет посмотреть, что и как пользователи ищут через наш модуль поиска.



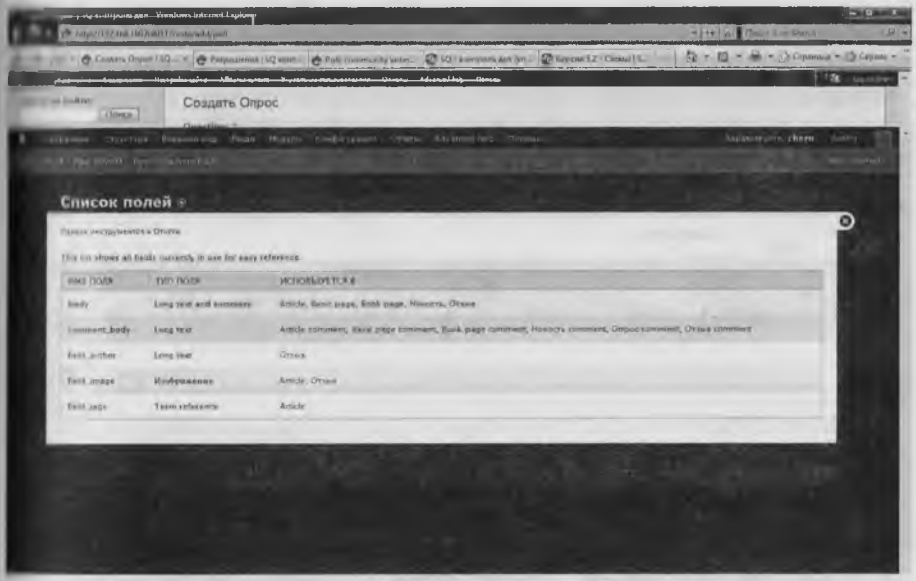
Почему в этот отчет попадают ссылки на картинки, которые никто не искал, непонятно.

Следующий отчет — **Последние посещения**.



Нельзя сказать, что это очень полезный отчет, однако он также показывает неудачные попытки запроса к страницам. Допустим, обращение к форуму, который еще не подключен, регистрируется в данном отчете со статусом **Страница не найдена**. В результате мы также можем делать выводы о работоспособности из данного отчета. Но если запросов окажется слишком много, данный отчет особо не поможет.

Далее следует отчет **Список полей**.

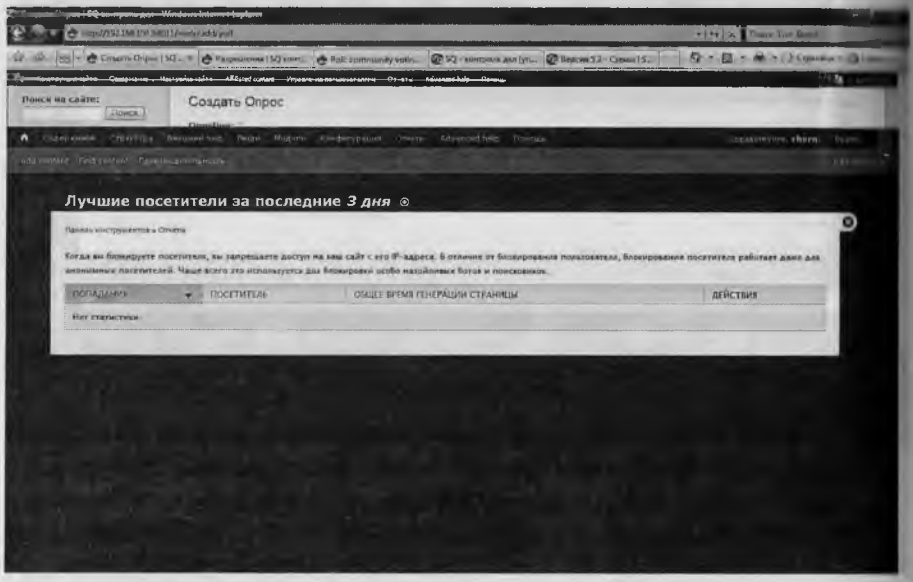


Чисто служебный полезный отчет, который группирует поля по отношению к типам материалов.

Следующий отчет, **Популярные страницы**, показывает наиболее популярные страницы, в том числе и те, которые не сработали. Заодно можно посмотреть время создания страниц.

Отчет **Лучшие посетители** позволяет посмотреть, с каких IP и кто заходил на наш сайт.

Проблема тут в том, что считаются не только прямые линки на страницы, но и открытия картинок и т. п.



Однако посетителю можно запретить доступ прямо из данного отчета.

Поскольку мы только что включили данный модуль, многие отчеты пусты, и по ним можно только сделать вывод, какие поля будут заполняться позже. Ничего, когда сайт заработает, все сразу появится, даже больше чем нужно.

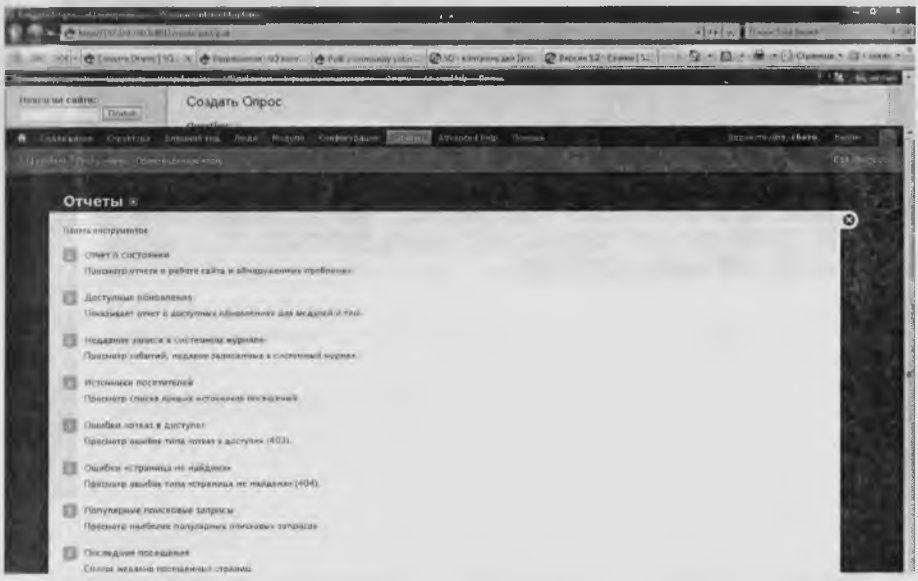
В общем, надо сказать, что статистика подойдет для настройки сайта, но не для серьезного анализа посетителей.

Надо либо искать более мощный модуль, либо устанавливать какой-либо из счетчиков.

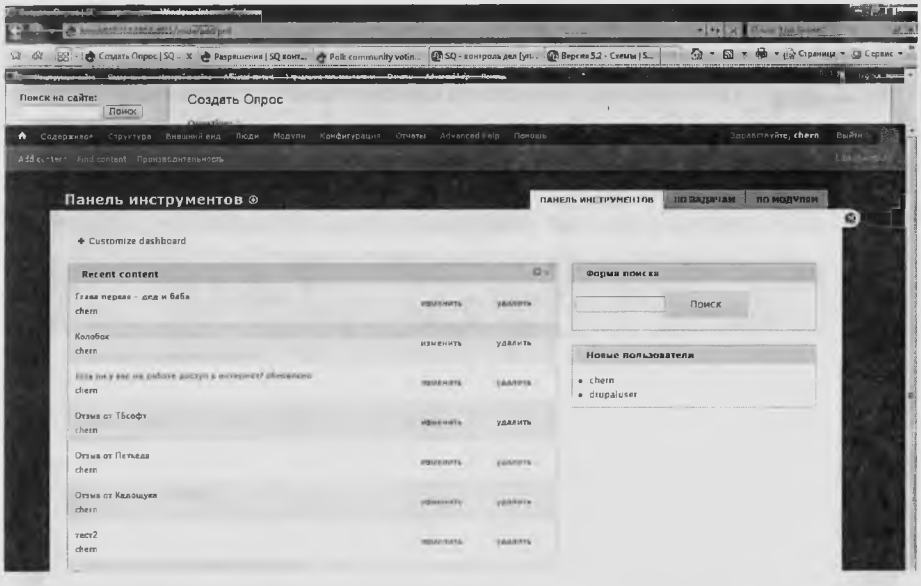
В Drupal 7 появилась новая возможность — **Панель инструментов** (Dashboard).

Мы можем указать на ссылку — найти ее непросто, она показана отдельным мелким текстом в административных меню.

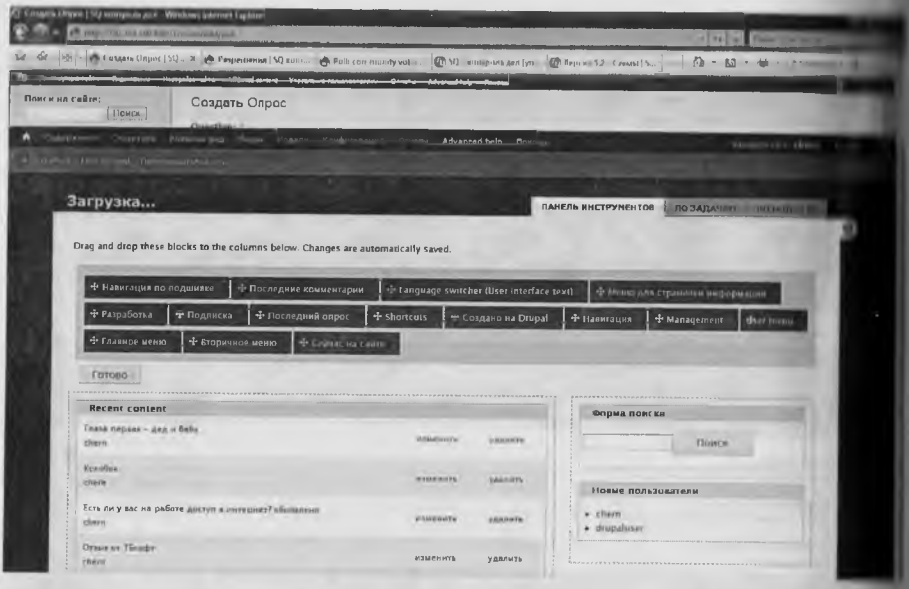
Однако ссылку можно добавить в строку меню.



Указав на нее, мы откроем стандартную панель.



Самое полезное — это, конечно, кастомизация.



Мы можем вывести только данные, которые интересуют именно нас, и сохранить внешний вид панели. После этого появится возможность увидеть ключевую информацию, которая нас интересует.

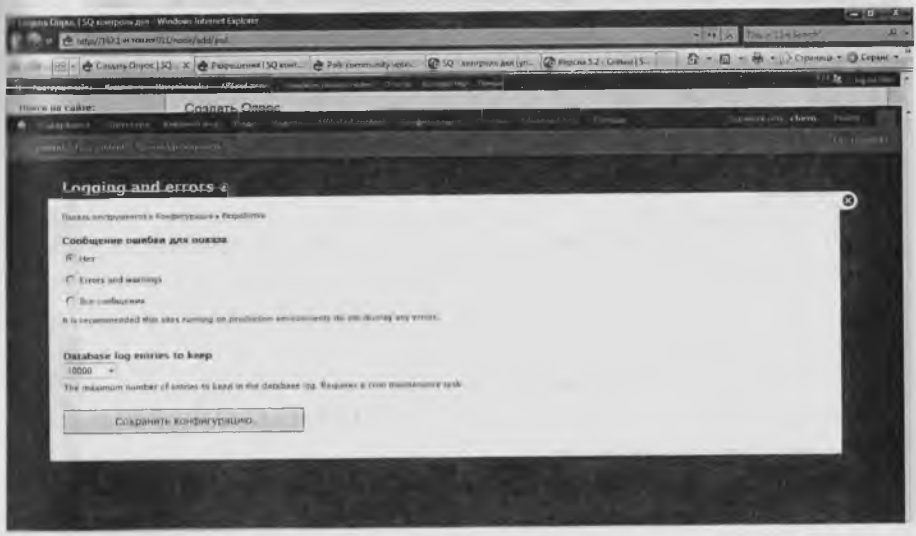
ПРИМЕЧАНИЕ

Почитайте также http://drupal.org/node/627162 о мониторинге сайта.

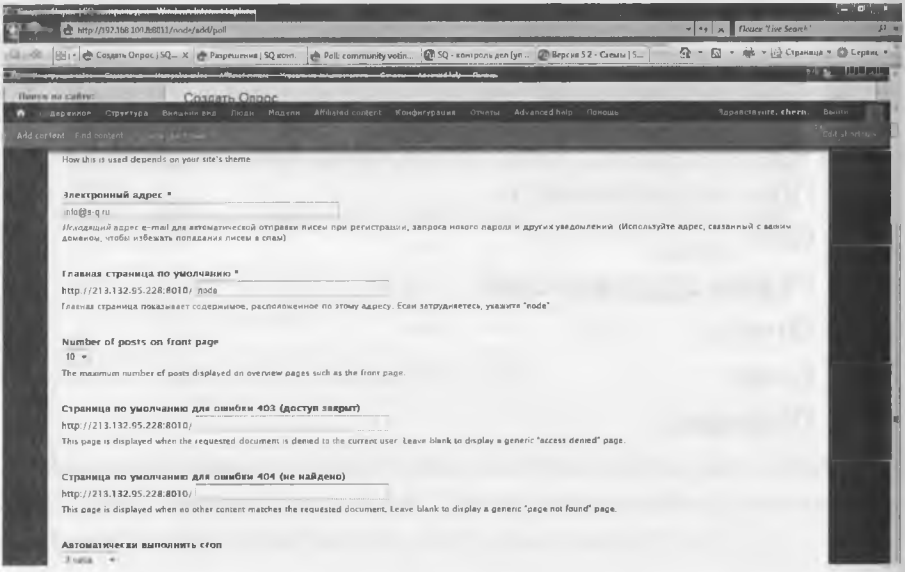
ПЕРЕВОД САЙТА В РАБОЧИЙ РЕЖИМ

- Logging and errors
- О сайте
- Производительность
- Cron
- Режим обслуживания
- Отчеты
- Бэкап
- Проверка

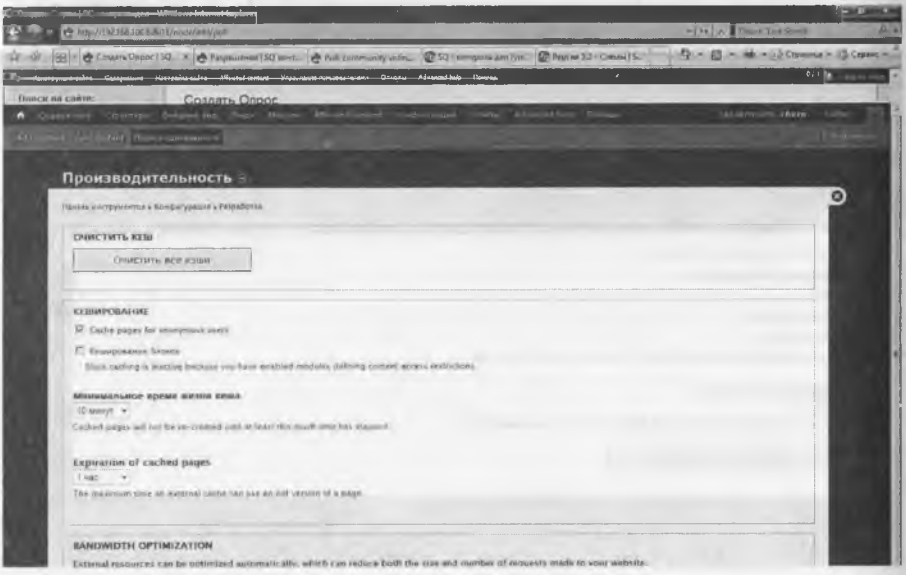
Давайте вкратце опишем действия, необходимые при переводе сайта в рабочий режим. Для начала надо провести ряд настроек. Выполняем команду **К ⇒ Разработка ⇒ Logging and errors** и указываем, что ошибки не отображаются (также можно поставить размер лога).



На странице **К ⇒ Система ⇒ О сайте** убеждаемся, что корректно прописана главная страница и есть странички для обработки ошибок 403 и 404 (их иметь желательно, но не обязательно — Drupal сам покажет свою стандартную страницу).



На странице **К ⇒ Разработка ⇒ Производительность** нужно включить все кэши и оптимизации.



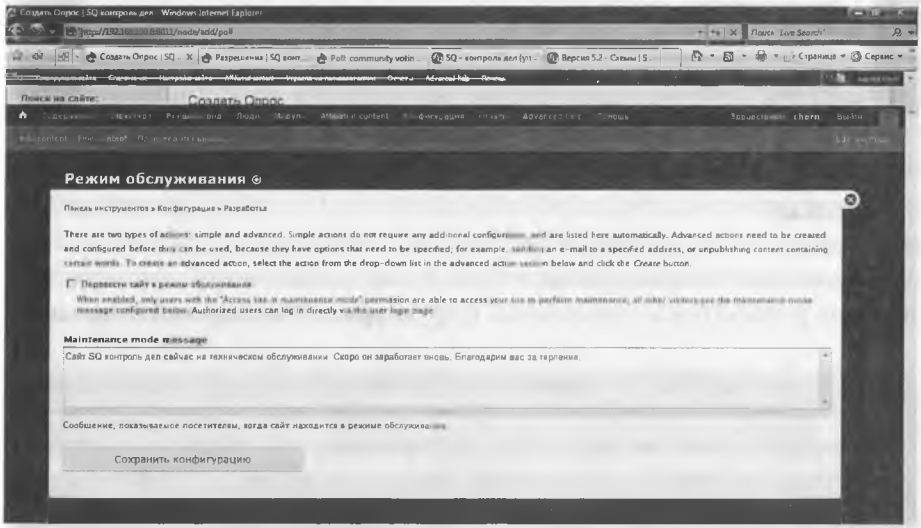
Необходимо корректно настроить cron, как описано в http://drupal.org/getting-started/6/install/cron. Он выполняет различные периодические задачи, в частности:

- проверку наличия апдейтов;
- реиндексацию базы данных для поиска;
- периодические бэкапы и т. д.

В принципе основные системные настройки на этом заканчиваются, необходимо также проводить регулярные операции, описанные ниже.

Перевод в режим обслуживания

Осуществляется в меню **К ⇒ Разработка ⇒ Режим обслуживания**.

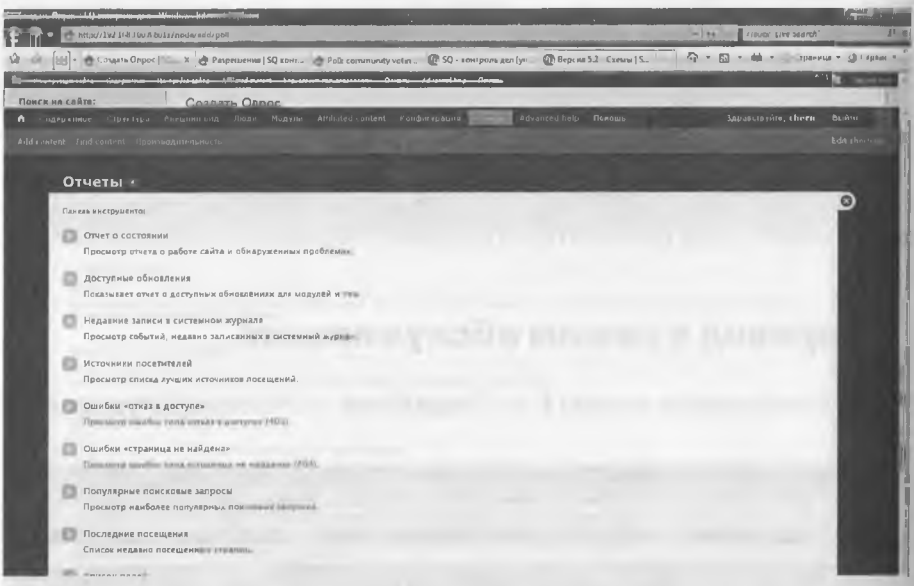


Все серьезные операции на сайте, например, ввод новых тем, инсталляция новых или апгрейд старых модулей, должны выполняться с переводом сайта в режим обслуживания.

Проверка отчетов

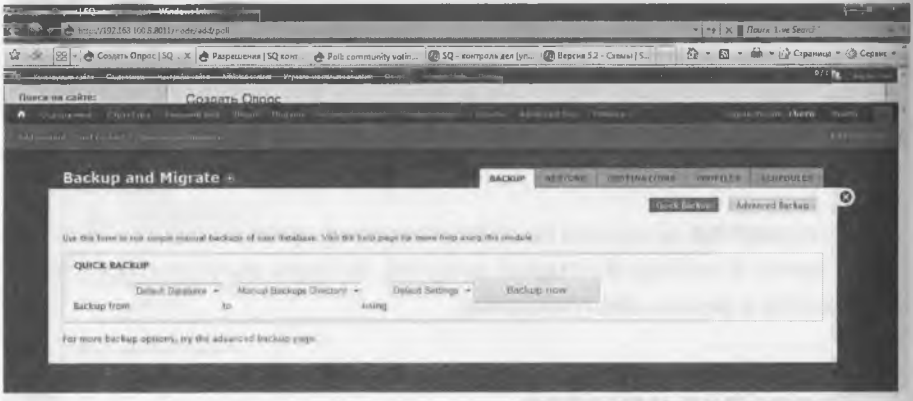
Как было описано в предыдущей главе, мы должны регулярно просматривать отчеты по работе сайта, исправлять ошибки и выполнять

нужные апдейты. Это делается в меню **Отчеты**. Также можно создать себе **Панель управления**.



Бэкап

Нужно настроить периодический бэкап, либо самому регулярно выполнять ручной бэкап через меню **Структура** ⇒ **Backup and Migrate**.



После этого вы можете переводить свой сайт в рабочий режим и показывать его посетителям. Естественно, не забудьте протестировать сайт для различных ролей пользователей — не только администратором.

МУЛЬТИСАЙТОВОСТЬ

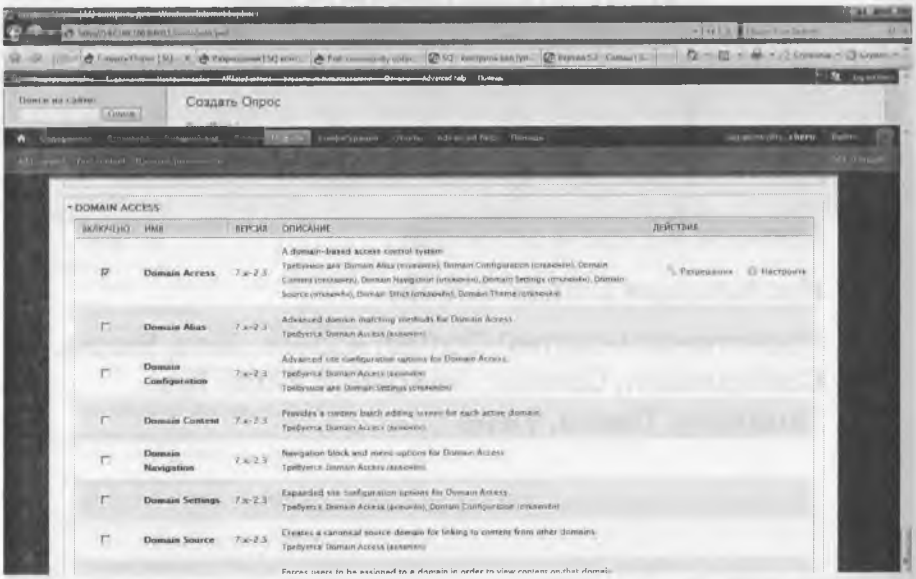
- Разные варианты мультисайтовости
- Domains Access и его части
- Инсталляция Domain Access
- Проявления модулей Domain Access, Alias, Blocks, Configuration, Content, Navigation, Node Type, Taxonomy, Theme, Views

Одна из проблем, которую мы должны были решить, заключалась в наличии нескольких сайтов. На Joomla необходимо покупать специальные модули, чтобы обеспечить поддержку нескольких сайтов одним приложением. На Drupal эта опция встроена изначально, и все ориентировано именно на несколько сайтов и возможность делить материалы между ними. В данном случае под сайтами мы понимаем сайты с различными доменными именами.

На самом деле есть разные варианты мультисайтовости. Например:

1. Много сайтов, для каждого — отдельная установка Drupal. Это, по сути, не мультисайтовость, а отдельные сайты.
2. Много сайтов, установка Drupal одна, но на каждый сайт своя база данных — это именно то, что обычно понимают под мультисайтовостью. Достигается такая структура путем инсталляции нескольких баз данных и файлов settings.php. Плюс тут в том, что обновлять нужно всего одну версию Drupal. В остальном сайты полностью независимы.
3. Много сайтов, но и установка Drupal одна, и база данных одна. Удобно тем, что можно разделять контент между сайтами. Это наш вариант, поскольку все наши сайты тесно взаимосвязаны. Реализуется такая структура при помощи модуля **Domain Access** (http://drupal.org/project/domain).

Модуль включает множество частей:



- **Domain Access** — базовый модуль — он нам нужен;
- **Domain Alias** — позволяет создавать синонимы и шаблоны имен доменов для создания соответствия множества имен одному имени домена — полезный;
- **Domain Configuration** — позволяет задавать такие системные переменные, как домашняя страница, статус автономной работы и т. п., раздельно для каждого субдомена — нужно;
- **Domain Content** — предоставляет различные административные страницы материалов для каждого субдомена так, что редакторы могут администрировать материалы своих секций сайта — очень полезен;
- **Domain Navigation** — предоставляет навигационный блок с тремя темами и создает раздельные пункты меню для каждого субдомена — нужен (у нас явно различные меню на разных сайтах);
- **Domain Prefix** — предоставляет пользовательский интерфейс выбора префикса базы данных для использования в субдоменах — мы можем сделать так, что часть таблиц будет разделяться для наших модулей, а часть останется общей. Потом в этом никогда не разобраться — не будем включать — пусть все остается в одной куче.

(Внимание! На момент написания в версии для Drupal 7 отсутствовал! Возможно, функции поменяются!);

- **Domain Source** — позволяет редакторам установить первичный домен, когда линки указывают с других доменов — в файле README.txt задана область применения — нам не нужно (если вы используете SEO-оптимизацию в модуле **Domain access**, то лучше его включить);
- **Domain Strict** — приписывает пользователей к доменам для просмотра содержимого — пока не нужно;
- **Domain Theme** — позволяет задавать темы, их настройки и цвета для каждого субдомена — включаем;
- **Domain User** — позволяет автоматически создавать субдомены для зарегистрированных пользователей — это нужно при необходимости создавать субдомены примерно как на сайте narod.ru — нам не нужно. (Внимание! На момент написания в версии для Drupal 7 отсутствовал! Возможно, функции поменяются!);
- **Domain Views** — предоставляет интеграцию **View** с доменами — включаем. (Внимание! На момент написания в версии для Drupal 7 отсутствовал! Возможно, функции поменяются!).

Также вам могут понадобиться некоторые модули из наследников. Мы опишем несколько полезных модулей:

- **Domain Blocks** (http://drupal.org/project/domain_blocks) — позволяет задавать различный внешний вид блоков в разных субдоменах — сайты имеют различное оформление — нужен;
- **Domain CTools** (http://drupal.org/project/domain_ctools) — позволяет задавать принадлежность к доменам для страниц и панелей — полезен;
- **Domain Node Type** (http://drupal.org/project/domain_nodetype) — можно задавать домен по умолчанию для типа материала — мы изначально часть типов материалов делали для первого сайта, и они не должны присутствовать в остальных — нужен;
- **Domain Taxonomy** (http://drupal.org/project/domain_taxonomy) — добавляет признаки домена к терминам таксономии — зачем нам лишние термины на других сайтах?

Возможно, вам понадобятся и другие наследники. Все они приведены на странице описания модуля **Domain Access**, а на момент написа-

ния отсутствовали в Drupal 7. Будьте внимательны — функции могут измениться!

Для установки **Domain Access** необходимо прочитать INSTALL.QUICKSTART.txt. В общем, все сводится к следующему:

1. Добавляем в settings.php строки:

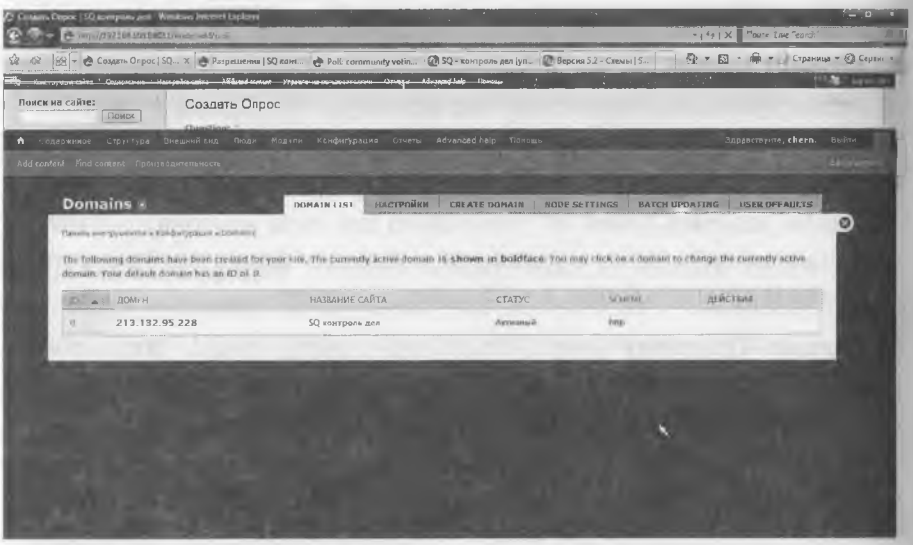
```
/**
 * Add the domain module setup routine.
 */
include DRUPAL_ROOT . '/sites/all/modules/domain/
settings.inc';
```

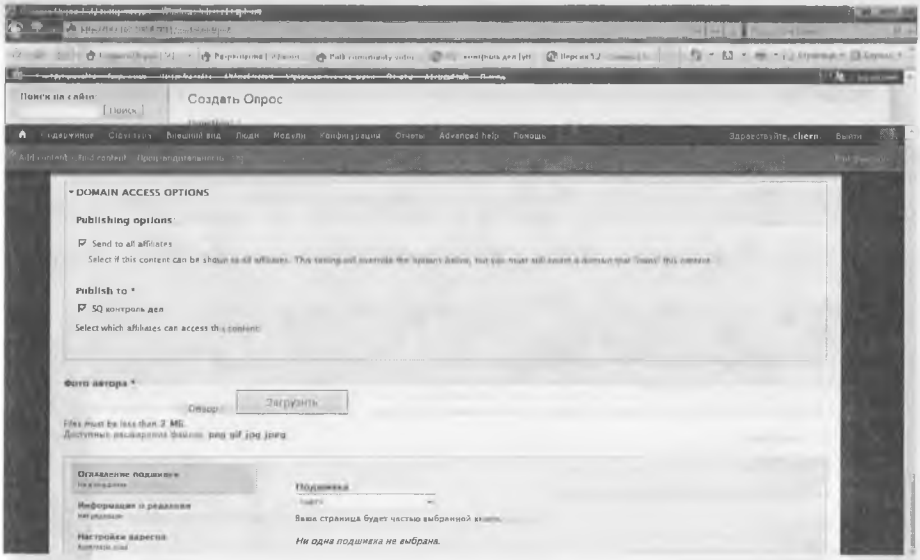
2. Включаем **Domain Access**.
3. Перестраиваем права доступа — из **Отчета о состоянии** или через линк после установки модуля.

Дальше можно включать остальные модули и настраивать домены в **К ⇒ Domains**.

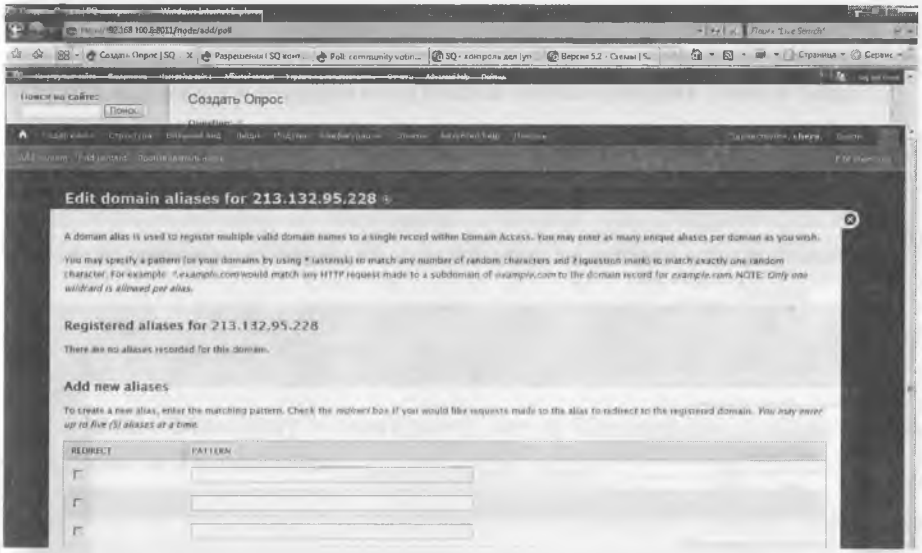
Давайте разберем проявления каждого из модулей, которые мы включили:

- **Domain Access** — появляется пункт меню **К ⇒ Domains**, где можно создавать список доменов и имена сайтов для них. Для материала добавляется опция принадлежности к домену;



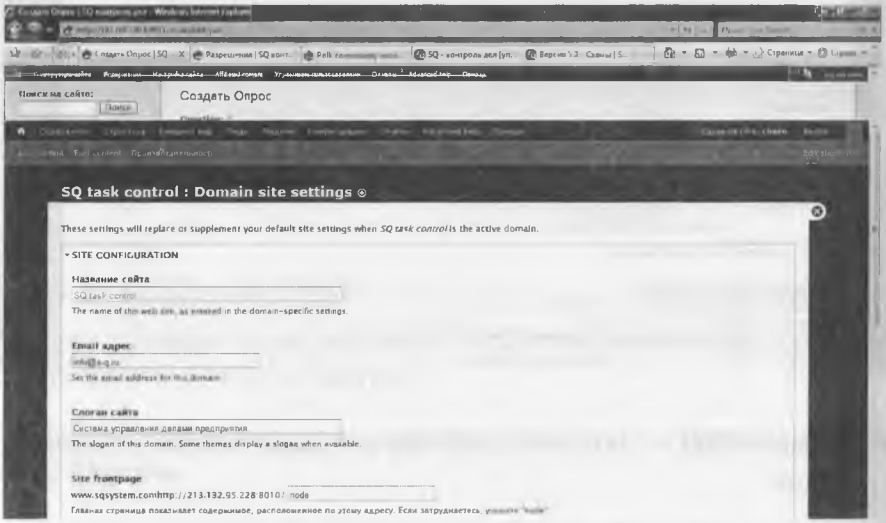


- **Domain Alias** — добавляет действия **add alias** и **aliases** в списке доменов;

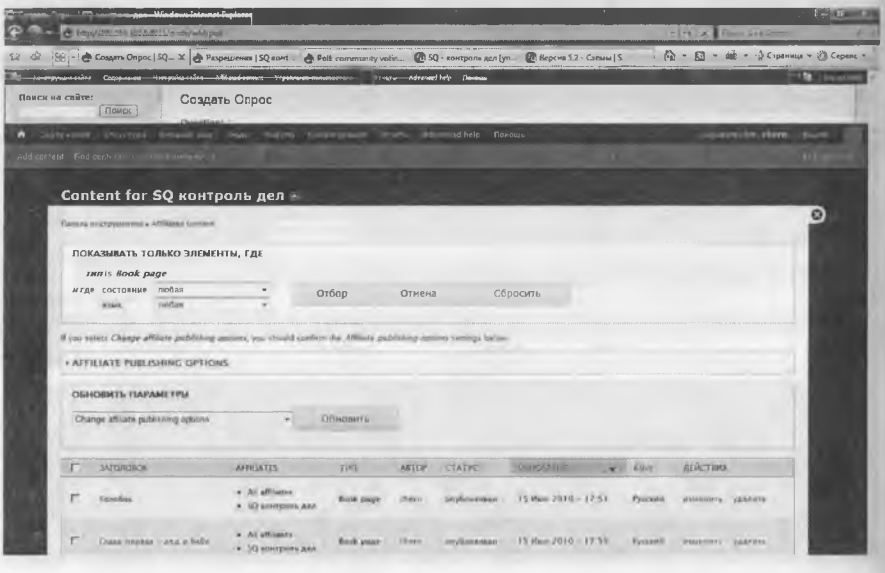


- **Domain Blocks** — в настройках для каждого блока появляется раздел **Domain specific visibility settings**. В нем можно указать, в каких доменах его нужно показывать. Также добавляются дополнительные полезные блоки **Domain Switcher** и **Domain Access Information**;

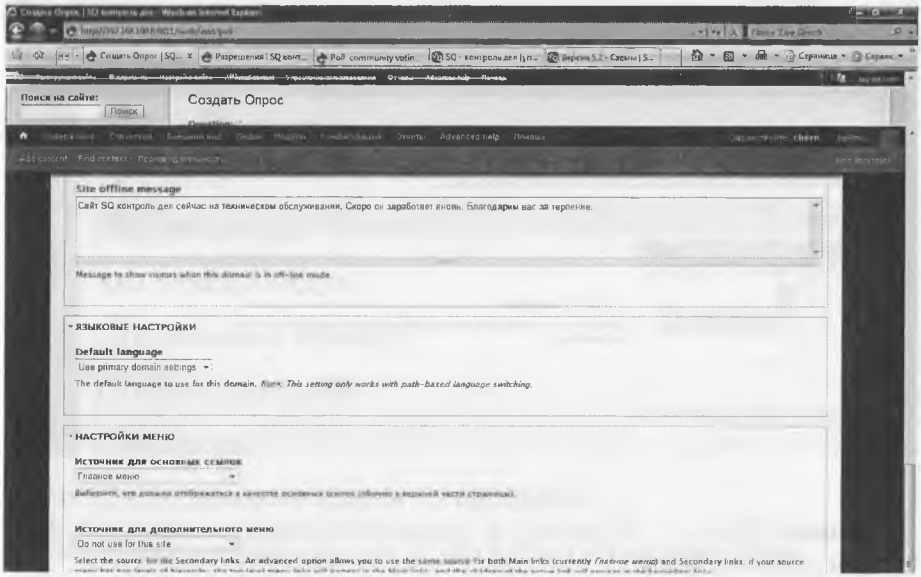
- **Domain Configuration** — в списке **A ⇒ Конструкция сайта ⇒ Domains ⇒ Domain list** для вторичных доменов появляется действие **settings**. Открыв его, можно настроить параметры вторичного сайта и статус онлайн/автономная работа;



- **Domain Content** — появляется новое меню **Affiliated content**, в котором можно выбрать сайт и посмотреть контент только для данного сайта.



Тот же выбор можно сделать из общего списка сайтов, после чего откроется следующее окно:



В нем также можно осуществлять групповые операции по привязке материалов к сайтам;

- **Domain Navigation** — в настройках домена (действие **settings**) появляется раздел **Настройки меню**.

  В нем можно выбрать меню по умолчанию для каждого из доменов-сайтов;

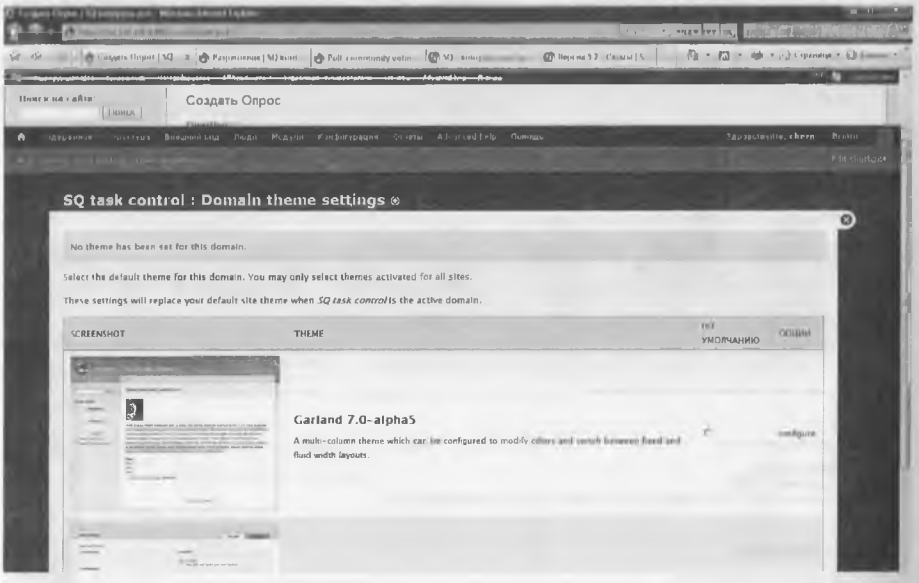
- **Domain Node Type** — для каждого типа материала можно задать сайт, на котором он должен быть опубликован;

- **Domain Taxonomy** — появляется меню **Конструкция сайта** ⇒ **Domains** ⇒ **Таксономия**.

Включив в словаре правила доступа к доменам, для его терминов можно задавать правила публикации в различных доменах;

- **Domain Theme** — в списке доменов появляется действие **Тема оформления**.

  Мы можем выбрать, какие темы используем, и детально задать их настройки для каждого домена;



- **Domain Views** — добавляется возможность фильтрации по типу домена внутри **View**.

При помощи указанных модулей мы легко можем создать и настроить несколько сайтов с разным оформлением и единым контентом. Главное, чтобы все доменные имена указывали на основной сервер, где стоит Drupal.

МУЛЬТИТЕМНОСТЬ И МНОГОЯЗЫЧНОСТЬ

- Sections
- Задание тем
- Многоязычность
- Добавление языков
- Content translation
- Переводы
- Определение языка интерфейса
- Дополнительные модули **Internationalization** и **translation management**

Одна из проблем, которая не была решена при помощи мультисайтовости, состоит в следующем: у нас четыре домена, но сайтов больше. То есть структура сайтов следующая:

- www.s-q.ru — основной сайт SQ контроль дел;
- www.sqsystem.com — английский сайт SQ контроль дел;
- www.s-q.ru/constructor — сайт продукта SQ конструктор;
- www.s-q.ru/affiliate — сайт для партнеров и т. д.

При этом каждый из сайтов имеет свою тему оформления. При помощи мультисайтовости можно задавать разные темы для разных доменов, но нам нужно большее.

Используя сайт drupal.org, находим, что данную проблему можно решить при помощи модуля **Sections** (http://drupal.org/project/sections). Кто бы мог подумать, что он так называется! К сожалению, данный модуль тоже еще не готов для Drupal 7.

Модуль позволяет определять секции (по начальному пути) и задавать для каждой из них новую тему. Немного непонятно, как это все будет работать с мультидоменностью, но думаем, что настройка будет действовать на все домены разом. Поскольку сейчас пути во всех наших доменах едины, мы не сможем завести два сайта вида:

- www.s-q.ru/constructor — русский сайт продукта SQ конструктор;
- www.sqsystem.com/constructor — английский сайт SQ конструктор.

Но для этого вполне можно использовать следующие пути:

- www.s-q.ru/constructor — русский сайт продукта SQ конструктор;
- www.sqsystem.com/en/constructor — английский сайт SQ конструктор;
- www.sqsystem.com/de/constructor — немецкий сайт SQ конструктор.

В данном модуле есть одна проблема. В результате его действия получается один сайт, который отражается в нескольких темах. С точки зрения Drupal, это означает, что параметры сайта можно задать только для домена. Для сайта в поддиректории мы не можем задать таких же параметров, как для отдельного сайта. Также проблема существует с работой в меню и т. п. Однако мы уже умеем писать сниппеты, поэтому сумеем соорудить программку, которая возьмет то, что нам нужно, и откуда нужно...

Кстати, модуль **Sections** также дает возможность задать тему для каждого конкретного материала при его создании и редактировании.

Раз уж мы заговорили про другие языки, то давайте разберемся, каким образом вообще включается многоязычность. Мы предполагаем два варианта многоязычности.

1. Два разных домена для сайтов на разных языках, например:

   www.s-q.ru/constructor — русский сайт продукта SQ конструктор;

   www.sqsystem.com/en/constructor — английский сайт SQ конструктор.

2. Язык различается по коду в URL, например:

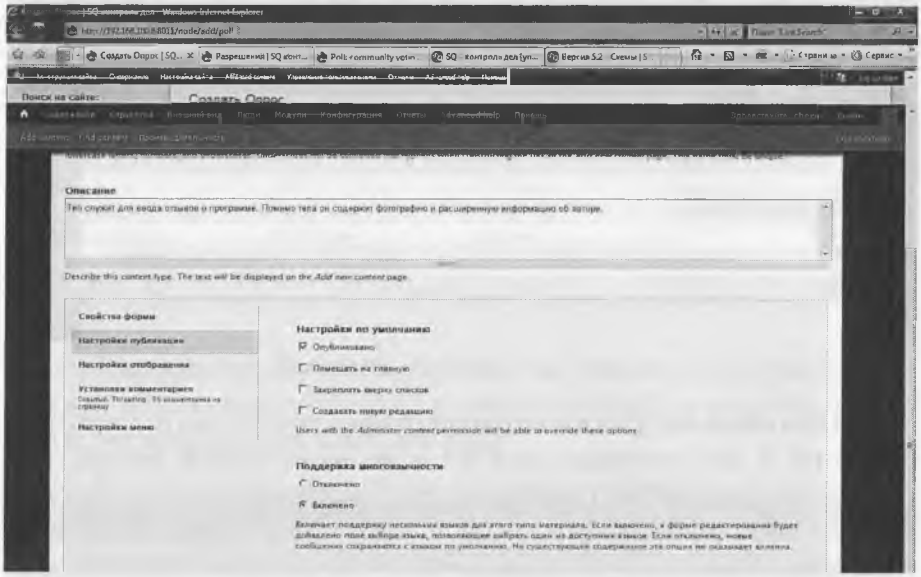
   www.sqsystem.com/en/constructor — английский сайт SQ конструктор;

   www.sqsystem.com/de/constructor — немецкий сайт SQ конструктор.

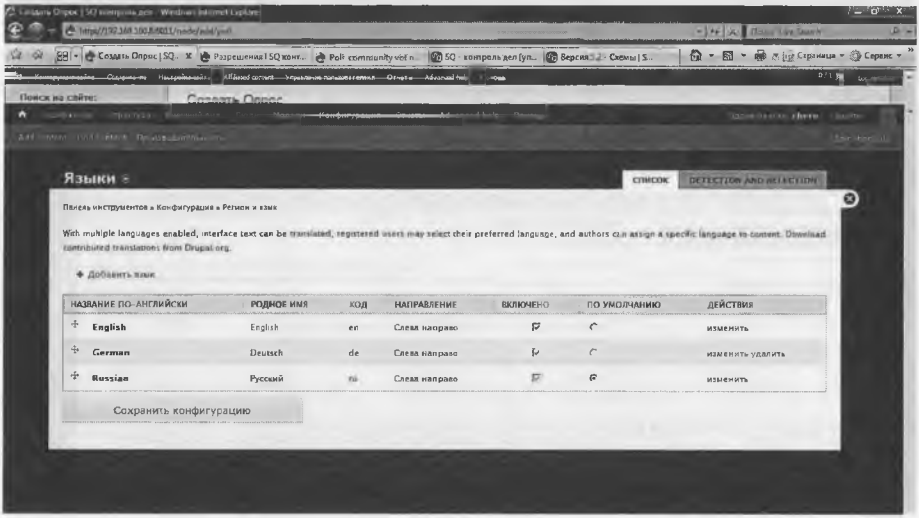
Оба варианта часто встречаются в жизни, поэтому мы должны разобраться, как их реализовывать...

Для начала нужно указать, для каких типов материалов мы хотим использовать многоязычность. Используем наш многострадальный тип **Отзыв** и начнем экспериментировать с ним.

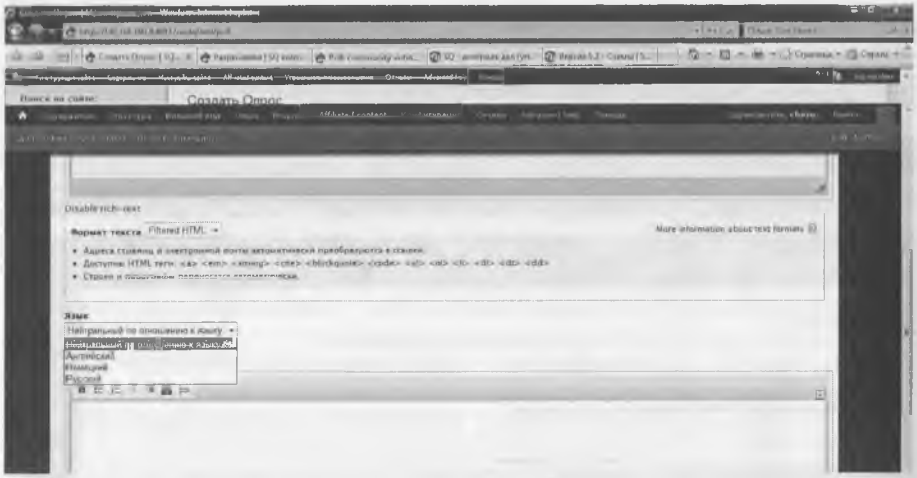
Сначала укажем, что данный тип является многоязычным.



Далее проверим список доступных языков (**К ⇒ Регион и язык ⇒ Языки**). Добавим немецкий.

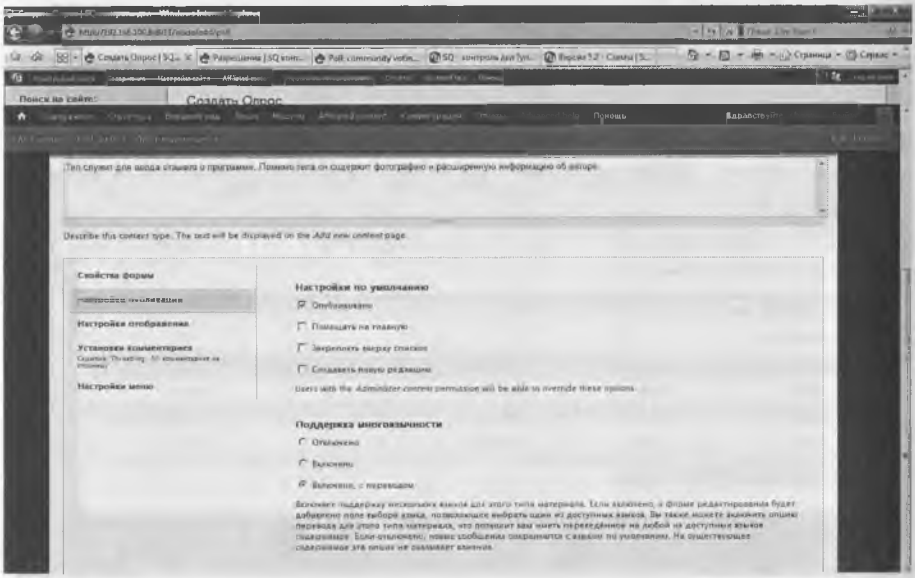


После этого при вводе материалов появится опция языка.

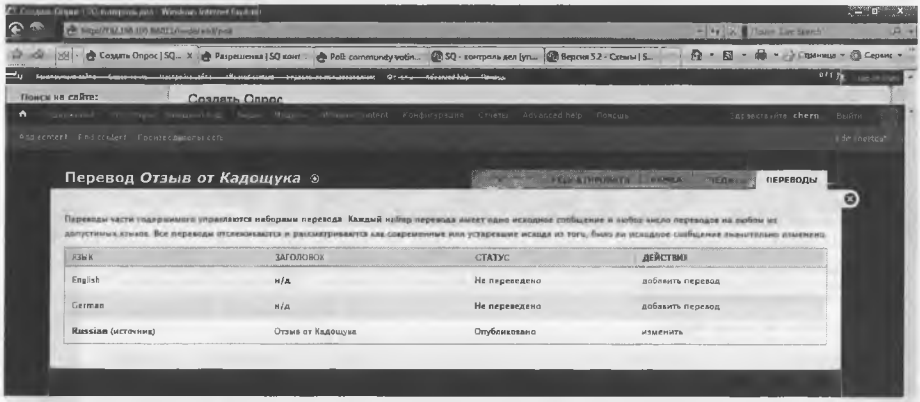


Но для ввода материалов это нужно лишь в том случае, если они различаются. А если материал один и тот же, но на разных языках, пригодится модуль **Content translation**. Это стандартный модуль, который входит в ядро. Включаем его.

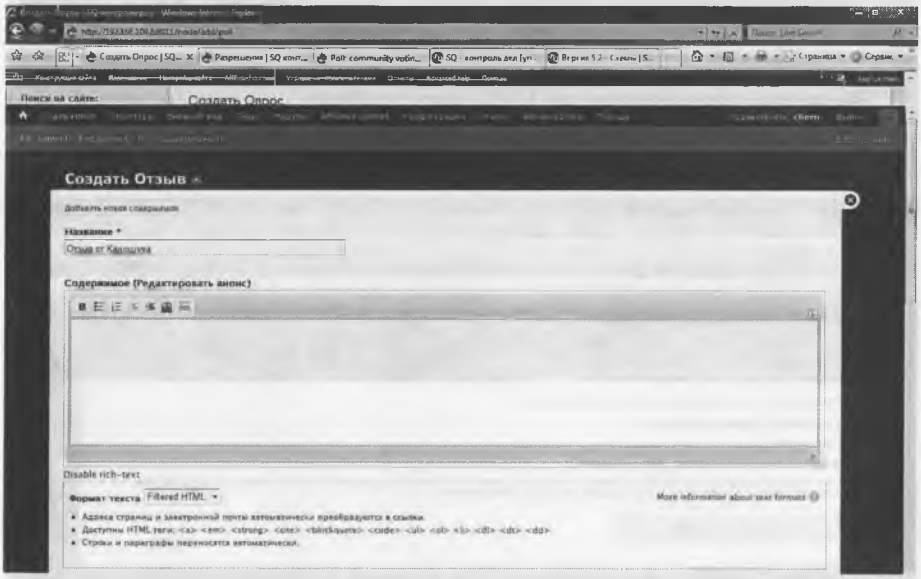
Теперь появляется дополнительная опция для типа материала — **Включено, с переводом**. Выберем ее.



После этого при редактировании материала отображается новая закладка **Переводы**. Выберем и изучим ее.

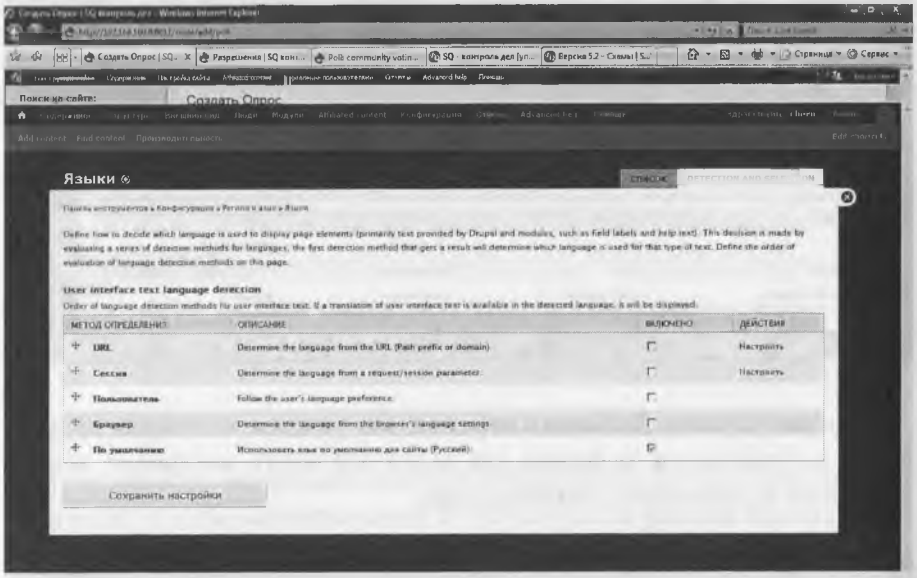


Выбрав вариант **Добавить перевод**, мы увидим, что экран практически не отличается от экрана для ввода основного материала, только пропадает возможность поменять язык.

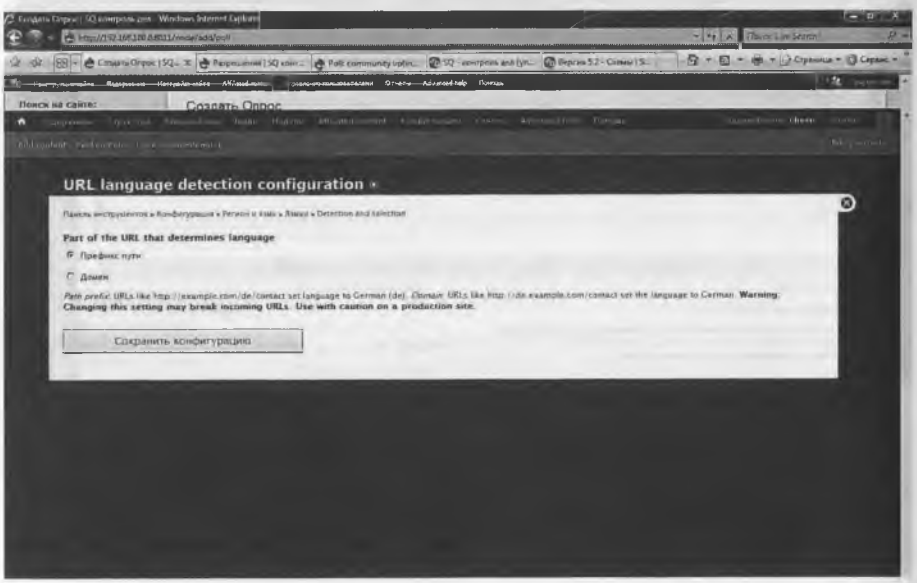


При создании перевода все равно создается новая страница, и смысл только в том, что внутри страницы связаны между собой, и мы можем легко отслеживать, какие изменения в переведенных страницах нужно произвести при изменении основного материала.

Для настройки способа, которым выбирается текущий язык интерфейса, служит меню **К ⇒ Регион и язык ⇒ Языки ⇒ Detection and Selection**.



Включив определение по URL в настройках, мы видим, что можем использовать только один из вариантов для определения языка.



То есть стандартными методами мы получаем выбор языка либо на основе префикса пути, либо на основе домена. Но все методы определения применяются поочередно. Если система не сможет определить язык на основе первого метода, то она возьмет следующий включенный, а это по умолчанию русский. Так мы добьемся своей цели.

Вообще многоязычность — это довольно сложная задача. Наши сайты редко полностью повторяют друг друга, а это означает, что нужно иметь различную навигацию для разных сайтов и т. д. То есть при помощи выбора языка мы легко сможем вносить изменения лишь в текстовую часть тем, а еще надо менять картинки и т. п. В результате часто оказывается, что проще сделать два сайта на разных доменах с различной навигацией, темами и т. д.

Однако, если вы предполагаете наличие большого объема информации, то стоит изучить полностью вопрос «правильной» многоязычности. Для этого рекомендуется использовать дополнительные модули **Internationalization** (http://drupal.org/project/i18n) и **Translation Management** (http://drupal.org/project/translation_management). В Drupal 7 их пока нет, но как же без них! Будем ждать...

ВНЕШНИЕ ПОЛЬЗОВАТЕЛИ И КОММЕНТАРИИ

- Настройки регистрации пользователей
- Разрешения
- Модуль Tracker

Ну и последняя тема, на которой хотелось бы остановиться, — это тема многопользовательской работы в Drupal. Имеется в виду не просто возможность ведения сайта несколькими пользователями, а поддержку внешних пользователей, что позволит создать сообщество пользователей сайта с различными функциями и, может быть, даже ролями.

Для этого нужно разрешить таким пользователям регистрироваться на сайте и позволить им вводить информацию. Чтобы разрешить пользователям регистрироваться, необходимо выполнить команду **К ⇒ Люди ⇒ Настройки учетной записи**.

Здесь мы и настраиваем регистрацию пользователей, поведение при их удалении, а также форматы писем, которые будут приходить пользователям при различных операциях. Политику и настройку этой секции мы диктовать не будем — каждый выбирает ее самостоятельно.

Определив вашу политику и введя нужные настройки, необходимо вернуть блок **Вход в систему**, поместить его в определенное место вашего сайта и оформить в соответствии с темой аналогично тому, как мы меняли представление блока поиска.

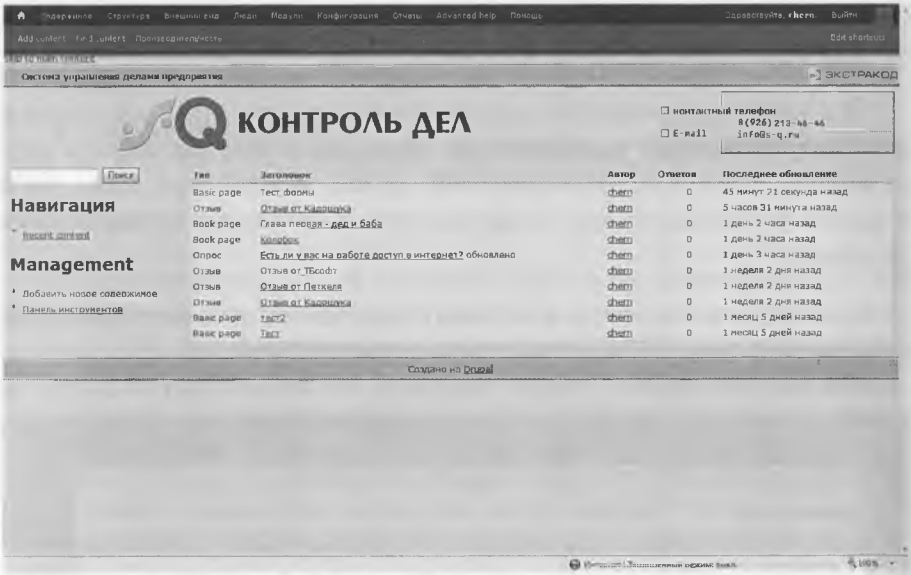
Все зарегистрированные пользователи получают права из колонки **зарегистрированные пользователи**. Если вы хотите расширить возможности некоторых пользователей, нужно завести еще одну роль и давать ее пользователям по результатам выполнения определенных условий.

Для начала пользователям следует разрешить оставлять на сайте свои комментарии. Переходим в директорию разрешений.

Здесь мы прописали жесткую политику, запретив пользователям отправлять комментарии без одобрения. Это означает, что все комментарии подтверждаются администратором, прежде чем стать достоянием общественности. Такую политику вы можете изменить по своему усмотрению. Еще мы разрешили читать комментарии.

Для удобного просмотра и управления комментариями нужно включить стандартный модуль **Tracker**. Он входит в ядро Drupal.

После включения данного модуля в меню **Navigation** (помните, мы писали, что в него информацию добавляют модули) появится ссылка на новый отчет.



Мы можем посмотреть, кто, когда и что вводил в систему, и принять какие-либо меры. Или не принимать их — все-таки сайт для людей, а не люди для сайта.

ЗАКЛЮЧЕНИЕ

В целом, Drupal 7 оставляет двойственное впечатление. С одной стороны, вроде все стало лучше и красивее, с другой — кое-где либо недо-, либо переусердствовали.

Например:

- административное меню переместилось наверх, есть модуль **overlay**, но меню не иерархическое, и пока доберешься до нужного пункта, потеряешь уйму времени и откроешь множество страниц. Особенно это раздражает, когда что-то меняешь в глубине в редактировании, а затем хочешь вернуть назад, а ни **breadcrumb**, ни кнопки **Cancel** нет. Надеюсь, что это был баг, и в официальной версии появится какой-то вариант;
- еще больше увеличилась путаница со стилями: теперь две темы — административная и сайта — должны жить на одном экране в режиме оверлей;
- непонятно, зачем в названиях шаблонов вставили двойные минусы. Теперь надо постоянно думать, где его вставить;
- количество возможных опций возросло, хотя работа с ними организована более удобно, особенно при вводе материалов.

Книга писалась, когда часть модулей еще отсутствовала. Некоторые модули были упомянуты, некоторые, например **FAQ** (http://drupal.org/project/faq), остались неохваченными. Вообще модули взаимосвязаны так, что мы идем по цепочке от одного к другому и так можем не остановиться в поиске полезных «штучек», которые нам никогда не понадобятся. В случае с Drupal 7 цепочка пока быстро прекращается, поскольку модули еще не готовы, но это все вопрос времени.

Некоторые модули мы вынуждены были править, причем возникали очевидные синтаксические ошибки. Иногда модули, например **Pathauto**, приходилось собирать из двух версий, чтобы добиться корректной работы.

Тем не менее все трудности преодолены, и я надеюсь, что книга не сильно отойдет от действительности к моменту издания, поможет новичкам освоить Drupal 7 и активно применить свои знания на практике.

- Page.tpl.php
- Node.tpl.php
- Block.tpl.php
- Comment.tpl.php
- Шаблоны-кандидаты

Все описания переменных скопированы с сайта http://api.drupal.org и просто переведены на русский язык. Возможно, в исходном варианте там будет более актуальная или более полная информация.

Предопределенные переменные для файла page.tpl.php

Переменные общего назначения:

- `$base_path` : — URL-путь к директории Drupal. Обычно, как минимум, равен значению по умолчанию, равному /;
- `$directory` : — директория, в которой находится шаблон, например modules/system или themes/garland. Используется для формирования пути к картинкам темы;
- `$is_front` : — равно TRUE, если мы на главной странице сайта;
- `$logged_in` : — равно TRUE, если пользователь зарегистрирован и вошел в систему;
- `$is_admin` : — равно TRUE, если пользователь является администратором.

Идентификация сайта:

- `$front_page` : — URL главной страницы сайта. Необходимо использовать данную переменную, а не `$base_path` для формирования ссылок на нее. Включает `domain` или `prefix` языка;

- `$logo` : — путь к картинке логотипа, как определено в конфигурации темы;
- `$site_name` : — название сайта. Переменная пуста, если показ названия отключен в настройках темы;
- `$site_slogan` : — слоган сайта, пуст, если показ слогана отключен.

Навигация:

- `$main_menu (array)` : — массив, который содержит первичные навигационные линки для сайта, если они были сконфигурированы;
- `$secondary_menu (array)` : — массив, который содержит вторичные навигационные линки для сайта, если они были сконфигурированы;
- `$breadcrumb` : — Breadcrumb для текущей страницы.

Содержание страницы (в порядке вхождения в стандартном файле page.tpl.php):

- `$title_prefix (array)` : — массив дополнительных префиксов к заголовку страницы, генерируемый модулями;
- `$title` : — заголовок страницы для использования внутри HTML-содержания;
- `$title_suffix (array)` : — массив дополнительных суффиксов к заголовку страницы, генерируемый модулями;
- `$messages` : — HTML для статуса и сообщений об ошибках. Должен быть ярко выделен;
- `$tabs` : — закладки, которые ведут на подстраницы для данной страницы (например, закладки просмотра и редактирования при показе материала);
- `$action_links (array)` : — локальные для страницы действия, например **Add menu** в интерфейсе администрирования меню;
- `$feed_icons` : — строка с иконками для текущей страницы;
- `$node` : — объект с содержимым материала, если у вас есть материал, который соответствует данной странице, либо если идентификатор материала является вторым аргументом в пути к странице (то есть объект будет загружен для страниц node/12345 и node/12345/revisions, но не для comment/reply/12345).

Регионы:

- `$page['help']` : — динамический текст помощи для страниц администрирования;
- `$page['highlight']` : — элементы региона выделенного содержимого;
- `$page['content']` : — основное содержимое текущей страницы (собственно показываемый материал);
- `$page['sidebar_first']` : — элементы для первого бокового региона (левая колонка);
- `$page['sidebar_second']` : —элементы для второго бокового региона (правая колонка);
- `$page['header']` : — элементы заголовка;
- `$page['footer']` : — элементы завершающего региона.

Предопределенные переменные для файла node.tpl.php

Доступные переменные:

- `$title` : — отфильтрованный (без HTML-тегов) заголовок материала;
- `$content` : — массив элементов материала. Для вывода всего материала нужно вызвать `render($content)`, а для вывода части материала вызывается `render($content['field_example'])`. Для временного запрещения печати элемента нужно вызвать `hide($content['field_example'])`;
- `$user_picture` : — картинка автора материала из user-picture.tpl.php;
- `$date` : — форматированная дата создания (используйте `$created` для переформатирования при помощи `format_date()`);
- `$name` : — отформатированное имя автора материала, производимое функцией `theme_username()`;
- `$node_url` : — URL страницы текущего материала;
- `$display_submitted` : — отображение информации об авторе и дате создания материала;

- `$classes` : — строка классов, которая может быть использована для форматирования при помощи CSS. Мы можем манипулировать данными классами при помощи переменной `$classes_array` в функции препроцессирования.

  По умолчанию классы формируются следующим образом:

  - `node`: — текущий тип шаблона, то есть возможность перехвата на уровне темы,
  - `node-[type]`: — текущий тип материала. Например, если материал — «Blog entry», это может быть «node-blog». Заметим, что `type` — это машинное имя, и оно часто бывает короткой формой того названия, которое видно пользователям,
  - `node-teaser`: — материалы в форме анонса,
  - `node-preview`: — материалы в форме предпросмотра.

  Следующие классы управляются опциями публикации материала:

  - `node-promoted`: — материалы, опубликованные на центральной странице,
  - `node-sticky`: — материалы, которые закреплены вверху списка анонсов,
  - `node-unpublished`: — неопубликованные материалы, которые видны только администраторам;
- `$title_prefix (array)` : — массив дополнительных префиксов к заголовку страницы, генерируемый модулями;
- `$title_suffix (array)` : — массив дополнительных суффиксов к заголовку страницы, генерируемый модулями.

Другие переменные:

- `$node` : — объект материала. Содержит всю информацию о материале, но может включать небезопасные данные;
- `$type` : — тип материала, то есть story, page, blog и т. д.;
- `$comment_count` : — число комментариев к материалу;
- `$uid` : — идентификатор автора материала;
- `$created` : — время, когда материал был опубликован отформатированный как Unix timestamp;

- `$classes_array` : — массив значений атрибутов HTML-классов. Он собирается в строку, которая лежит в переменной `$classes`;
- `$zebra` : — признак четности. Может принимать значения «even» (четный) или «odd» (нечетный). Полезен для форматирования списков анонсов в виде зебры;
- `$id` : — порядковый номер материала. Инкрементируется каждый раз при выводе.

Переменные статуса материала:

- `$view_mode` : — режим показа, например `'full'`, `'teaser'`;
- `$teaser` : — TRUE, если выводится анонс (более короткое написание для выражения `$view_mode == 'teaser'`);
- `$page` : — TRUE, если выводится в полноэкранном режиме;
- `$promote` : — TRUE, если это вывод анонса материала на главной странице;
- `$sticky` : — TRUE, если материал закреплен вверху списка;
- `$status` : — TRUE, если материал опубликован;
- `$comment` : — режим комментариев: 0 — отключены, 1 — только чтение, 2 — запись и чтение;
- `$readmore` : — TRUE, если анонс не может включить весь материал и дается в сокращенном виде;
- `$is_front` : — TRUE, если идет показ на главной странице;
- `$logged_in` : — TRUE, если пользователь зарегистрирован и вошел в систему;
- `$is_admin` : — TRUE, если пользователь является администратором.

Переменные для полей: для каждого поля, которое присутствует в материале, определяется соответствующая переменная, например для `$node->body` определяется переменная `$body`. При необходимости обратиться к неформатированным значениям полей всем разработчикам крайне рекомендуется использовать именно эти переменные. В противном случае придется явно указать требуемый язык поля, например `$node->body['en']`, что полностью перекроет все ранее примененные правила для корректного определения языка.

Предопределенные переменные для файла block.tpl.php

Доступные переменные:

- `$block->subject` : — заголовок блока;
- `$content` : — содержимое блока;
- `$block->module` : — модуль, который произвел данный блок;
- `$block->delta` : — численный идентификатор модуля;
- `$block->region` : — регион, в который встроен данный блок;
- `$classes` : — строка классов, которая может быть использована для форматирования при помощи CSS. Мы можем манипулировать данными классами при помощи переменной `$classes_array` в функции препроцессирования. По умолчанию классы формируются следующим образом:
  - `block`: — текущий тип шаблона, то есть возможность перехвата на уровне темы,
  - `block-[module]`: — модуль, который сгенерировал данный блок. Например, модуль **User** отвечает за работу стандартного блока навигации для пользователей. В этом случае класс — `block-user`;
- `$title_prefix (array)` : — массив дополнительных префиксов к заголовку страницы, генерируемый модулями;
- `$title_suffix (array)` : — массив дополнительных суффиксов к заголовку страницы, генерируемый модулями.

Вспомогательные переменные:

- `$classes_array` : — массив значений атрибутов HTML-классов. Он собирается в строку, которая лежит в переменной `$classes`;
- `$block_zebra` : — принимает значения `'odd'` и `'even'` внутри региона блока;
- `$zebra` : — принимает значения `'odd'` и `'even'` внутри страницы (вне региона блока);
- `$block_id` : — счетчик, который перечисляет блок внутри региона блоков;

- `$id` : — счетчик, который перечисляет блок внутри страницы (вне региона блока);
- `$is_front` : — TRUE, когда блок выводится на главной странице;
- `$logged_in` : — TRUE, если пользователь зарегистрирован и вошел в систему;
- `$is_admin` : — TRUE, если пользователь является администратором;
- `$block_html_id` : — корректный идентификатор внутри HTML, гарантированно уникальный.

Предопределенные переменные для файла comment.tpl.php

Доступные переменные:

- `$author` : — автор комментария. Может быть как линком, так и текстом;
- `$content` : — массив элементов комментария. Для вывода всего комментария нужно вызвать `render($content)`, а для вывода части комментария — `render($content['field_example'])`. Для временного запрещения печати элемента нужно вызвать `hide($content['field_example']);`
- `$created` : — отформатированная дата и время создания комментария. Функция препроцессирования может переформатировать ее при помощи `format_date()` с требуемыми параметрами, используя переменную `$comment->changed`;
- `$new` : — маркер нового комментария;
- `$permalink` : — постоянная ссылка на страницу с комментарием. Поясним: комментарии могут добавляться и удаляться достаточно динамично. Если мы выводим комментарии на нескольких страницах, то могут меняться параметры постраничного вывода. Данная ссылка позволяет при переходе по ней всегда попадать на страницу, где будет расположен данный комментарий при любом текущем способе вывода комментариев;
- `$picture` : — картинка автора;

- `$signature` : — подпись автора;
- `$status` : — статус комментария. Возможные значения: comment-unpublished (неопубликован), comment-published (опубликован) или comment-preview (предпросмотр);
- `$title` : — прилинкованный заголовок;
- `$classes` : — строка классов, которая может быть использована для форматирования при помощи CSS. Мы можем манипулировать данными классами при помощи переменной `$classes_array` в функции препроцессирования. По умолчанию классы формируются следующим образом:
  - `comment:` — текущий тип шаблона, то есть возможность перехвата на уровне темы,
  - `comment-by-anonymous:` — комментарий незарегистрированного пользователя,
  - `comment-by-node-author:` — комментарий автора материала-родителя,
  - `comment-preview:` — в режиме предпросмотра нового или редактируемого комментария.

  Следующие классы управляют показом для зарегистрированных пользователей;
  - `comment-unpublished:` — неопубликованный комментарий, видимый только администраторам,
  - `comment-by-viewer:` — комментарий пользователя, который сейчас просматривает страницу,
  - `comment-new:` — новый комментарий, появившийся после последнего посещения пользователем страницы;
- `$title_prefix (array)` : — массив дополнительных префиксов к заголовку страницы, генерируемый модулями;
- `$title_suffix (array)` : — массив дополнительных суффиксов к заголовку страницы, генерируемый модулями.

Следующие переменные определяют контекст:

- `$comment` : — полный объект комментариев;
- `$node` : — объект материала, к которому привязаны комментарии.

Вспомогательные переменные:

- `$classes_array` — массив значений атрибутов HTML-классов. Он собирается в строку, которая лежит в переменной `$classes`.

Шаблоны-кандидаты

Кандидаты для шаблонов — это альтернативные файлы шаблонов, которые переопределяют основной файл шаблона. Они работают только в той же директории, что и базовый шаблон (важно!). Это означает, например, что для корректной работы вашего специфического comment-blog.tpl.php необходимо создать также comment.tpl.php в вашей теме (оба файла должны быть в одной и той же директории).

В Drupal 7 зачем-то повставляли кое-где двойные минусы — будьте внимательны! И при переносе с Drupal 6 придется переименовывать файлы шаблонов.

Можно также создавать свои шаблоны-кандидаты путем переопределения функций preprocess:

`block--[region|[module|-delta]].tpl.php`

базовый шаблон: `block.tpl.php`

Кандидаты формируются в следующем порядке:

1. `block--module-delta.tpl.php`
2. `block--module.tpl.php`
3. `block--region.tpl.php`

`"module"` — это название модуля, а `"delta"` — внутренний идентификатор, который присвоен блоку модулем. Например, block--user-1.tpl.php используется для блока навигации пользователей, поскольку присутствует в модуле **user** под номером 1. `"region"` работает при использовании блока в некотором регионе.

`comment--[type].tpl.php`

базовый шаблон: `comment.tpl.php`

Здесь была встроена поддержка только файлов comment--*type*.tpl.php для форматирования комментариев определенного типа материала. Аналогично node--[type].tpl.php, но для комментариев:

comment-wrapper--[type].tpl.php

базовый шаблон: comment-wrapper.tpl.php

Аналогично предыдущему, но для форматирования шаблона wrapper — «обертки» для комментариев:

forums--[[container|topic]--forumID].tpl.php

базовый шаблон: forums.tpl.php

Кандидаты формируются в зависимости от следующих факторов в указанном порядке.

Для контейнера форума:

1. forums--containers--forumID.tpl.php
2. forums--forumID.tpl.php
3. forums--containers.tpl.php

Для топиков форума:

1. forums--topics--forumID.tpl.php
2. forums--forumID.tpl.php
3. forums--topics.tpl.php

maintenance-page--[offline].tpl.php

базовый шаблон: maintenance-page.tpl.php

Применяется при проблемах с базой данных. Полезен для показа приветливых страниц без списка ошибок.

Нужно правильно сконфигурировать страницу, как описано в **Theming the maintenance page** (http://drupal.org/node/195435).

node--[type].tpl.php

базовый шаблон: node.tpl.php

Тип для показа материала определенного типа, например node--story.tpl.php, node--blog.tpl.php и т. п. Не забудьте, что для использования шаблонов конкретного типа необходимо обязательно переопределить основной node.tpl.php файл. Если его не будет, все файлы вида node--[type].tpl.php не будут замечены темой:

`page--[front|internal/path].tpl.php`

базовый шаблон: `page.tpl.php`

Кандидаты очень многочисленны. page--front.tpl.php используется для центральной страницы. Остальные базируются на пути к текущей странице. Не путайте внутренний путь с синонимом, синонимы здесь не вычисляются.

Центральная страница может быть установлена в **A ⇒ Настройка сайта ⇒ О сайте**. Она будет обрабатываться через page--front.tpl.php.

Список шаблонов-кандидатов в порядке предпочтения базируется на внутреннем пути. Для каждого элемента пути создается свой кандидат. Понять можно только на примере:

http://www.example.com/node/1/edit приводит к следующим шаблонам-кандидатам:

1. `page--node--edit.tpl.php`
2. `page--node--1.tpl.php`
3. `page--node.tpl.php`
4. `page.tpl.php`

`poll-results--[block].tpl.php`

базовый шаблон: `poll-results.tpl.php`

Функция, которая показывает результаты голосования, разделена между материалом и блоком. Базовый шаблон для материала, но можно использовать шаблон-кандидат для внутренности блока. Данный кандидат также присутствует в modules/poll/poll-results-block.tpl.php.

`poll-vote--[block].tpl.php`

базовый шаблон: `poll-vote.tpl.php`

Аналогично poll-results-[block].tpl.php, но для генерации форм для голосования. Необходим собственный шаблон для получения нужного эффекта.

`poll-bar--[block].tpl.php`

базовый шаблон: `poll-bar.tpl.php`

То же, что и poll-vote-[block].tpl.php, но для генерации индивидуальных строк голосования.

`profile-wrapper--[field].tpl.php`

базовый шаблон: `profile-wrapper.tpl.php`

Данный шаблон используется при просмотре страницы со списком членов. При просмотре с фильтром по некоторому полю можно создать кандидата с именем этого поля. Например, http://drupal.org/profile/country/Belgium может предложить в кандидаты profile-wrapper-country.tpl.php.

`search-results--[searchType].tpl.php`

базовый шаблон: `search-results.tpl.php`

search-results.tpl.php — это базовый шаблон для результатов поиска. В зависимости от типа поиска могут быть предложены различные шаблоны-кандидаты. Например,example.com/search/node/Search+Term приведет к кандидату search-results-node.tpl.php. Сравните с example.com/search/user/bob, который приведет к search-results-user.tpl.php. Модули могут расширять типы поиска, добавляя новые шаблоны-кандидаты.

`search-result--[searchType].tpl.php`

базовый шаблон: `search-result.tpl.php`

То же, что и выше, только для результатов поиска.

ПРИЛОЖЕНИЯ ДЛЯ РАЗРАБОТЧИКА

- Как писать тексты
- Вывод полей в PHP
- Сниппеты
- Использование CVS

В довольно обширной документации на сайте drupal.org описано, как лучше оформлять программные тексты, писать различные конструкции в PHP и т. д. Вы можете обратиться по ссылке http://drupal.org/coding-standards и получить развернутую информацию. Мы же постараемся здесь дать минимальные знания для новичка.

В различных файлах шаблонов часто указывается наличие переменных `$page` или `$content`. Это объекты, которые имеют довольно сложную структуру. Мы можем воспользоваться следующим кусочком кода для показа его на нашей странице:

```
<?php print '<pre>'. check_plain(print_r($variable,
1)) .'</pre>'; ?>
```

Выделенную `$variable` нужно заменить вашей переменной. Далее уже по данной структуре можно понять, как добраться до той или иной информации.

Подобные полезные кусочки кода в Drupal называются сниппетами.

Сниппеты — это небольшие фрагменты кода, которые выполняют настолько простую задачу, что создавать для этого модуль нет смысла. Мы также приводили пример сниппета в главе «Таксономия». Он позволял раскрашивать по-разному страницу в зависимости от привязанного к материалу термина таксономии.

На сайте drupal.org есть отдельный раздел, посвященный сниппетам. Пользователи, которые посчитали, что их код кому-то может понадобиться, размещают их там. Сниппеты разбиты по темам, но можно пытаться искать их и контекстным поиском (как на сайте, так и в Интернете). Воспользуйтесь также адресом: drupal.ru.

Заметим, что нужно четко представлять себе, какую задачу вы решаете. Одна и та же задача может решаться как модулем, так и сниппетом. Модули обычно более объемные и общие, а сниппеты решают конкретную задачу, и часто приходится корректировать их текст, чтобы адаптировать к собственному сайту. Ну и, конечно, надо знать основы PHP.

В общем, решать вам. Понятно, что чем более нестандартный сайт вы делаете, тем больше сниппетов придется вставлять и, возможно, создавать самим. Но известно, что за красоту и удобство стоит побороться. Опять же, по-другому к полноценному программированию на Drupal никак не перейти. Хотя сейчас это и не так часто нужно, как в Drupal 5.

Зайдите еще на сайт api.drupal.org. Многие сниппеты используют механизм hook для перехватывания различных событий Drupal, и на этом сайте вы найдете подробное его описание. Наша книжка для начинающих, поэтому мы не будем детально останавливаться на этой теме.

Для разработчика мы расскажем только, каким образом работать с CVS. Дело в том, что когда модулей много, апдейтить их вручную довольно неудобно. В Drupal 7 встроен специальный механизм автоматического апдейта модулей, но при этом останется проблема с установкой конкретного патча. Поэтому настоящие программисты используют CVS.

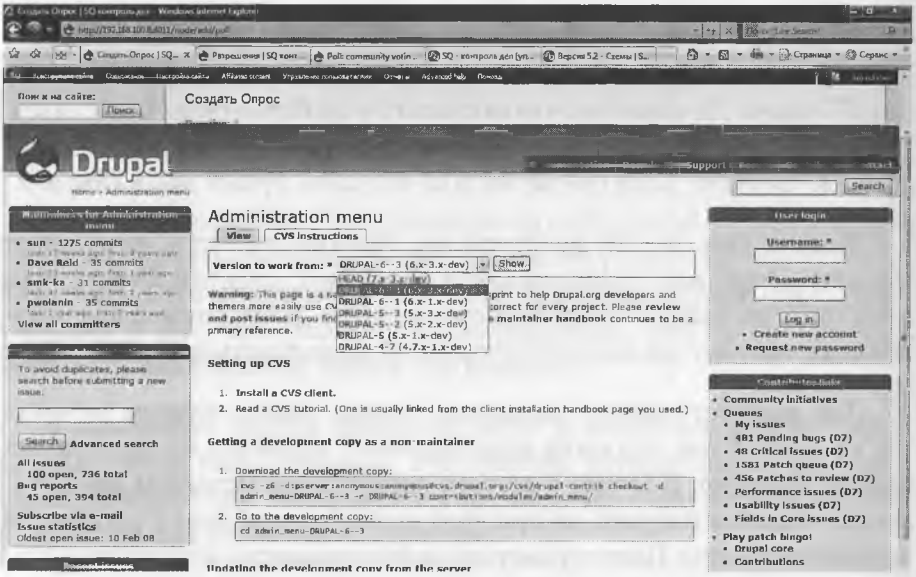
Также присутствует еще одна проблема — иногда новая версия модулей не содержит патчей, которые были выложены отдельно. Все эти проблемы могут быть решены путем перевода сайта на обновление модулей через CVS.

CVS (Concurrent Version System) — одна из наиболее распространенных систем управления версиями программных текстов. Основная идея ее заключается в том, что один и тот же текст программы могут править несколько человек, причем, если они правят в разных местах файла, слияние потом происходит совершенно незаметно.

В данном случае возникают исправления патчами, потом исправления вручную и исправления, связанные с выходом новой версии. Если мы установим данную систему, то до тех пор пока новая версия не за-

тронет фрагменты, исправленные вручную, никаких дополнительных усилий прикладывать не придется. Опять же, если мы настроим CVS, то загрузка новой версии модуля будет выполняться одной командой, и нам не придется сначала скачивать модуль, потом разархивировать его, копировать и устанавливать права доступа.

Каким же образом настраивается CVS? У всех модулей, которые выложены на drupal.org, есть отдельная закладка, в которой рассказано, как взять данный модуль из-под CVS. Например, для модуля **Administration menu**.



Соответственно, желая инсталлировать данный модуль, нужно набрать следующий код:

```
cvs -z6 -d:pserver:anonymous:anonymous@cvs.drupal.
org:/cvs/drupal-contrib checkout -d admin_menu-DRUPAL-
6--3 -r DRUPAL-6--3 contributions/modules/admin_menu/
```

Давайте разберем его подробно:

- `cvs` — название исполняемого модуля;
- `-z6` — уровень компрессии при передаче данных;
- `-d:pserver:anonymous:anonymous@cvs.drupal.org:/cvs/drupal-contrib` — адрес сайта;

- `checkout` — команда взятия содержимого модуля;
- `-d admin_menu-DRUPAL-6--3` — директория для помещения модуля (лучше написать `admin_menu`);
- `-r DRUPAL-6--3` — код релиза; берется из списка версий наверху;
- `contributions/modules/admin_menu` — директория на сервере, откуда снимаются данные.

Нам лучше сразу указывать для модулей конечную директорию admin_menu в sites/all/modules.

Тогда для взятия новой версии надо зайти в эту директорию и выполнить команду:

```
cvs update -dPr номер_версии
```

Например:

```
cvs update -dPr DRUPAL-6--3
cvs update -dPr DRUPAL-6--4
```

И т. д. по мере выхода версий.

Кстати, мы везде использовали

```
<?php
// $Id$
?>
```

Это стандартный идентификатор CVS. Если вы также поместите файлы тем под свой CVS, он заменится развернутой информацией о файле, дате создания и точной версии файла.

Дополнительную информацию о CVS читайте в http://drupal.org/handbook/cvs/quickstart.

Производственно-практическое издание

КОМПЬЮТЕР НА 100%

Черных Алексей Юрьевич

DRUPAL 7

Директор редакции *Л. Бершидский*
Ответственный редактор *В. Обручев*
Художественный редактор *Н. Биржаков*

ООО «Издательство «Эксмо»
127299, Москва, ул. Клары Цеткин, д. 18/5. Тел. 411-68-86, 956-39-21.
Home page: **www.eksmo.ru** E-mail: **info@eksmo.ru**

Подписано в печать 28.01.2011. Формат 70x100$^{1}/_{16}$.
Печать офсетная. Усл. печ. л. 16,85.
Тираж 2000 экз. Заказ № 972.

Отпечатано с готовых файлов заказчика в ОАО «ИПК
«Ульяновский Дом печати». 432980, г. Ульяновск, ул. Гончарова, 14

ISBN 978-5-699-47059-4

Оптовая торговля книгами «Эксмо»:
ООО «ТД «Эксмо». 142700, Московская обл., Ленинский р-н, г. Видное,
Белокаменное ш., д. 1, многоканальный тел. 411-50-74.
E-mail: **reception@eksmo-sale.ru**

По вопросам приобретения книг «Эксмо» зарубежными оптовыми покупателями *обращаться в отдел зарубежных продаж ТД «Эксмо»*
E-mail: **international@eksmo-sale.ru**

International Sales: *International wholesale customers should contact Foreign Sales Department of Trading House «Eksmo» for their orders.*
international@eksmo-sale.ru

По вопросам заказа книг корпоративным клиентам, в том числе в специальном оформлении,
обращаться по тел. 411-68-59, доб. 2115, 2117, 2118.
E-mail: **vipzakaz@eksmo.ru**

Оптовая торговля бумажно-беловыми и канцелярскими товарами для школы и офиса «Канц-Эксмо»:
Компания «Канц-Эксмо»: 142702, Московская обл., Ленинский р-н, г. Видное-2,
Белокаменное ш., д. 1, а/я 5. Тел./факс +7 (495) 745-28-87 (многоканальный).
e-mail: **kanc@eksmo-sale.ru**, сайт: **www.kanc-eksmo.ru**

Полный ассортимент книг издательства «Эксмо» для оптовых покупателей:
В Санкт-Петербурге: ООО СЗКО, пр-т Обуховской Обороны, д. 84Е.
Тел. (812) 365-46-03/04.
В Нижнем Новгороде: ООО ТД «Эксмо НН», ул. Маршала Воронова, д. 3.
Тел. (8312) 72-36-70.
В Казани: Филиал ООО «РДЦ-Самара», ул. Фрезерная, д. 5.
Тел. (843) 570-40-45/46.
В Ростове-на-Дону: ООО «РДЦ-Ростов», пр. Стачки, 243А.
Тел. (863) 220-19-34.
В Самаре: ООО «РДЦ-Самара», пр-т Кирова, д. 75/1, литера «Е».
Тел. (846) 269-66-70.
В Екатеринбурге: ООО «РДЦ-Екатеринбург», ул. Прибалтийская, д. 24а.
Тел. (343) 378-49-45.
В Новосибирске: ООО «РДЦ-Новосибирск», Комбинатский пер., д. 3.
Тел. +7 (383) 289-91-42. E-mail: **eksmo-nsk@yandex.ru**
В Киеве: ООО «РДЦ Эксмо-Украина», Московский пр-т, д. 9.
Тел./факс: (044) 495-79-80/81.
Во Львове: ТП ООО «Эксмо-Запад», ул. Бузкова, д. 2.
Тел./факс (032) 245-00-19.
В Симферополе: ООО «Эксмо-Крым», ул. Киевская, д. 153.
Тел./факс (0652) 22-90-03, 54-32-99.
В Казахстане: ТОО «РДЦ-Алматы», ул. Домбровского, д. 3а.
Тел./факс (727) 251-59-90/91. rdc-almaty@mail.ru

Полный ассортимент продукции издательства «Эксмо»
можно приобрести в магазинах «Новый книжный» и «Читай-город».
Телефон единой справочной: 8 (800) 444-8-444.
Звонок по России бесплатный.

В Санкт-Петербурге в сети магазинов «Буквоед»:
«Магазин на Невском», д. 13. Тел. (812) 310-22-44.

По вопросам размещения рекламы в книгах издательства «Эксмо» обращаться в рекламный отдел. Тел. 411-68-74.

Компьютер **на 100%**

Умные книги для умных людей!

Алексей Черных

Drupal 7

Бесплатная система управления сайтом

Drupal — это одна из самых популярных систем управления сайтом (CMS).

Будучи бесплатной и написанной на популярном языке PHP, Drupal получила самое широкое распространение и признание. Архитектура Drupal позволяет применять данную систему для построения любых сайтов — от интернет-магазинов до блогов, а также произвольным образом оформлять их.

Книга является простым и в то же время достаточно полным руководством по системе Drupal. Для изучения книги и последующего использования Drupal не требуется знания языков веб-программирования. Большое внимание уделено отличиям новой ве… системы от предыдущей.

ЭКСМО

ISBN 978-5-699-47059-4

9 785699 470594 >

www.education.eksmo.ru