ГЛАВА 10
Работа с блоками

Блоки — это фрагменты текста или функциональности, которые можно разместить в областях, определенных в шаблоне. Блоком может быть что угодно: одиночная нода, список нод, календарь, видеоклип, форма, онлайновый опрос, окно обмена сообщениями, поток изменений статусов в Facebook — в общем, практически все, что только можно себе представить. Когда речь идет о блоках, клиенты часто переспрашивают: "Значит, блок — это просто графический элемент?" Действительно, этот термин часто употребляется за пределами сферы Drupal для элементов, которые представляются в Drupal в виде блоков. В настоящей главе вы узнаете, как создавать собственные блоки и использовать их в своем сайте Drupal.

Что такое блок?

Как уже было сказано, блок — это обособленный контейнер, который может содержать практически все, что угодно. Проще всего понять, что такое блок, рассмотрев несколько примеров. В списке, приведенном в табл. 10.1, перечислены некоторые стандартные блоки, поставляемые вместе с Drupal.

Таблица 10.1. Стандартные блоки

Блок	Назначение
Login form (Входная форма)	Форма, отображаемая на сайте, которая позволяет пользователю выполнить вход, зарегистрировать новую учетную запись или изменить свой пароль
Who's online (Сейчас на сайте)	Блок со списком всех пользователей, которые в данный момент вошли на сайт
Who's new (Новые пользователи)	Блок со списком самых новых пользователей веб-сайта
Search form (Форма поиска)	Форма поиска, содержащаяся в блоке
Recent comments (Последние комментарии)	Список самых последних комментариев, отправленных на сайт
Main menu (Главное меню) и Secondary menu (Вторичное меню)	Оба меню доступны в виде блоков
Recent content (Последнее содержимое)	Список самых последних нод, отправленных на сайт
Most recent poll (Последний опрос)	Самый последний опрос на сайте отображается как блок (требуется активный модуль `Poll`)
Active forum topics (Активные темы форума)	Список тем форума, активных в последнее время (требуется активный модуль `Forum`)

Многие специальные модули включают блоки в качестве компонента обеспечиваемого ими функционального решения. Например, модуль `Ubercart` предоставляет несколько блоков для отображения такой информации, как состояние корзины посетителя.

Block API и интерфейс администрирования блоков позволяют создавать собственные блоки практически для любых целей. В табл. 10.2 приведены примеры специальных блоков, созданные автором за последнее время.

Таблица 10.2. Примеры специальных блоков

Блок	Назначение
Recent Bloggers (Последние блогеры)	Блок с галереей аватаров блогеров, которые разместили на сайте последние сообщения
Slideshow of upcoming events (Слайд-шоу из предстоящих событий)	Блок, отображающий тизеры нод для предстоящих событий в виде слайд-шоу
A chat form (Форма для чата)	Блок, который содержит графический элемент для чата Meebo и выводит этот элемент в левой или правой врезке
A donate now feature (Функция добровольного взноса)	Блок с кнопкой, которая позволяет пользователю внести взнос с помощью платежной системы PayPal
A list of new books added to a library's collection (Список новых книг, добавленных в библиотеку)	Блок с мини-изображением обложки книги, названием и автором книги и кнопкой "Зарезервировать"
A contact us form (Форма "Контакты")	Блок с простой формой запроса контактных данных
A list of postings on multiple social networking sites (Список сообщений на нескольких сайтах социальной сети)	Блок с лентами из нескольких сайтов социальной сети в виде общего смешанного списка
A Google map showing recent postings (Карта Google, показывающая последние сообщения)	Блок, выводящий маркеры на карте Google для нод, которые содержат географические координаты

Блоки определяются либо с помощью веб-интерфейса Drupal (рис. 10.1), либо программно через Block API (блоки, предоставляемые модулями). А как узнать, какой метод использовать для создания блока? Одноразовый блок вроде фрагмента статического HTML, относящегося к сайту — хороший кандидат на роль специального блока. Динамические по своей природе блоки, которые относятся к написанному вами модулю или состоят в основном из PHP-кода — хорошие кандидаты на применение Block API с реализацией внутри модуля. Старайтесь не держать PHP-код в собственных блоках, т.к. сопровождать код, хранящийся в базе данных, сложнее, чем код, размещенный в модуле. К тому же редактор сайта может нечаянно удалить результаты вашей напряженной работы. Если нет смысла создавать блок на уровне модуля, просто вставьте в блок вызов собственной функции и разместите весь PHP-код где-то в другом месте.

Совет. При необходимости создать специальные блоки или другие компоненты для конкретного сайта обычно лучше написать специальный модуль и поместить в него всю необходимую функциональность. Например, разработчик веб-сайта для компании Jones Pies and Soda может создать модуль `jonespiesandsoda`.

Интерфейс Block API использовать легко, однако это не значит, что он не позволяет создавать сложные вещи. Блоки могут отображать практически все (поскольку пишутся на PHP и поэтому не ограничены в своих возможностях), но они обычно играют вспомо-

гательную роль по отношению к основному контенту сайта. Например, можно создать специальные навигационные блоки для каждой пользовательской роли или блок для вывода списка комментариев, ожидающих утверждения.

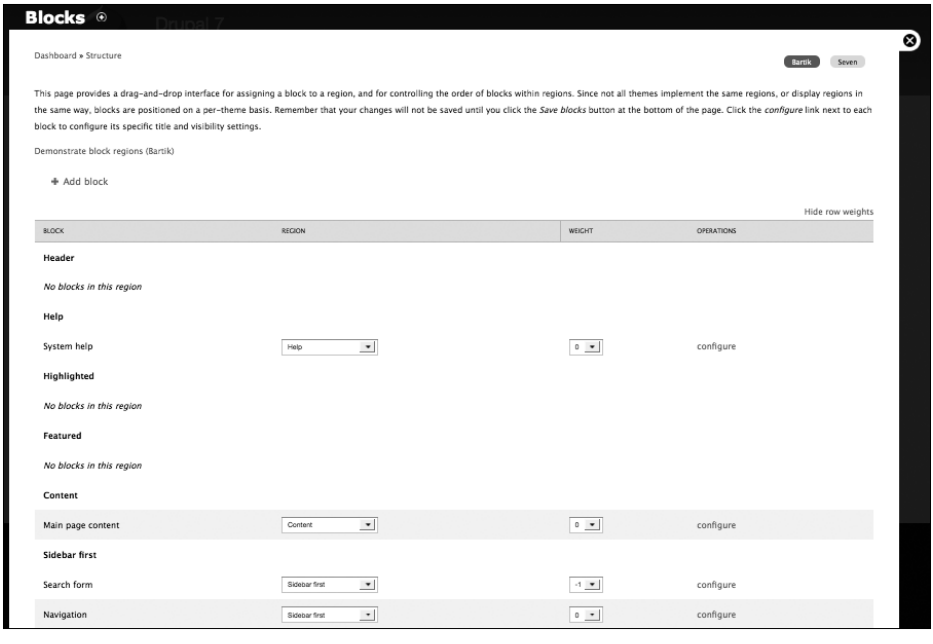

Рис. 10.1. Страница со списком блоков

Параметры настройки блоков

Часто применяемая настройка, с которой стоит внимательно ознакомиться — это параметры видимости блока на странице настройки этого блока. Видимость блока определяет, когда блок должен быть видимым на странице, а когда нет, в зависимости от критериев, заданных в интерфейсе, который показан на рис. 10.2. Рассмотрим в качестве примера блок `User Login` (Вход пользователя). Его отображением можно управлять с помощью перечисленных ниже параметров.

- *Параметры видимости на отдельных страницах.* Администраторы могут указать, что блок должен быть видимым или скрытым на конкретной странице или диапазоне страниц, или когда специальный фрагмент PHP-кода определит истинность каких-то условий.
- *Параметры видимости типов контента.* Администраторы могут указать, что блок должен быть видимым только на страницах, которые выводят конкретный тип контента — например, блок видим, только если страница содержит тему обсуждения.
- *Параметры видимости, зависящие от ролей.* Администраторы могут указать, что блок должен быть видимым только для пользователей с определенными ролями.
- *Параметры видимости, зависящие от пользователей.* Администраторы могут разрешить отдельным пользователям настраивать видимость некоторых блоков в параметрах учетных записей этих пользователей. Пользователи щелкают на ссылке My account (Моя учетная запись) и настраивают видимость блока.

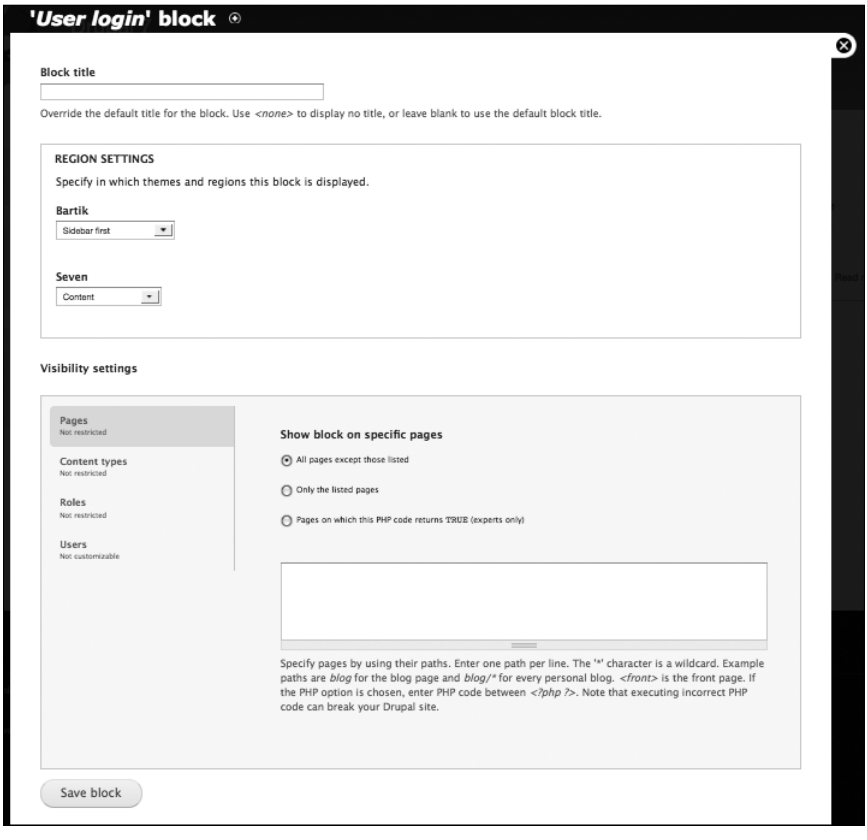

Рис. 10.2. Экран настройки блока в интерфейсе администрирования

Размещение блока

Выше уже было сказано, что страница администрирования блоков позволяет администратору сайта указать области, где могут появляться определенные блоки. На той же странице можно задать и порядок вывода блоков в области (см. рис. 10.1). Области определяются уровнем тем в `.info`-файле темы, а не с помощью Block API, и разные темы могут отображать различные области. Создание областей описано в главе 8.

Определение блока

Блоки определяются в модулях с помощью вызова `hook_block()`, и внутри одного этого хука модуль может реализовать несколько блоков. После определения блока он будет отображаться на странице администрирования блоков. Кроме того, администратор сайта может вручную создать специальные блоки, используя веб-интерфейс. В данном разделе основное внимание будет уделено программному созданию блоков. Посмотрим на схему базы данных для блоков, приведенную на рис. 10.3.

Свойства каждого блока хранятся в таблице `block`. Дополнительные данные для блоков, которые созданы с помощью интерфейса настройки блока, хранятся в других сопровождающих таблицах, также показанных на рис. 10.3.

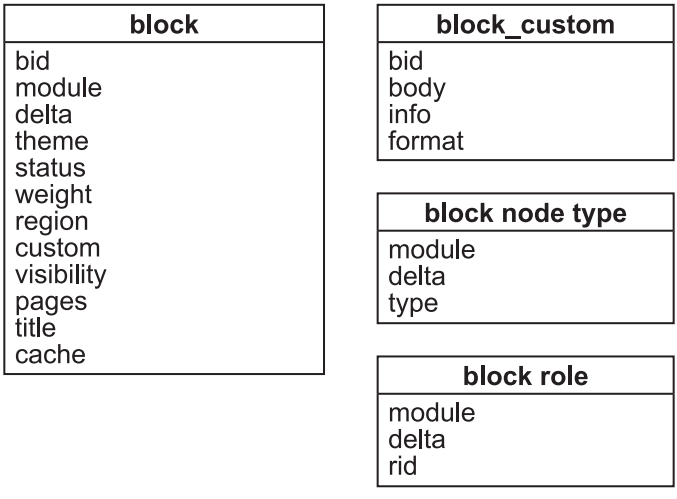

Рис. 10.3. Схема базы данных для блоков

Ниже перечислены свойства, определенные в столбцах таблицы `block`.

- `bid`. Уникальный идентификатор каждого блока.
- `module`. Содержит имя модуля, в котором определен данный блок. Блок входа пользователя создан в модуле `user` и т.д. Специализированные блоки, созданные администратором на странице Structure⇨Blocks⇨Add Blocks (Структура⇨Блоки⇨Добавить блоки), считаются созданными модулем `block`.
- `delta`. В одном вызове `hook_block()` модуль может определить несколько блоков, и поэтому столбец `delta` хранит ключ для каждого блока, уникальный лишь для каждой реализации `hook_block()`, но не для всех блоков вообще. Значение `delta` может быть целым числом или строкой.
- `theme`. Блоки можно определить для нескольких тем. Поэтому необходимо хранить имя темы, для которой активен блок. Для каждой такой темы в базе данных предусмотрена отдельная строка. Параметры конфигурации не распространяются на все темы.
- `status`. Содержит информацию о том, активирован ли блок. Значение `1` означает, что активирован, а значение `0` — отключен. Если с блоком не связана никакая область, Drupal заносит в это поле значение `0`.
- `weight`. Вес блока определяет его позицию относительно других блоков в области.
- `region`. Имя области, в которой должен появиться блок — например, `footer`.
- `custom`. Возможность настройки видимости данного блока пользователем (см. рис. 10.3). Значение `0` означает, что пользователь не может управлять видимостью данного блока; значение `1` — что блок по умолчанию видим, но пользователь может скрыть его; `2` — что блок по умолчанию скрыт, но пользователь может сделать его видимым.
- `visibility`. Определяет видимость блока. Значение `1` означает, что блок видим на всех страницах, кроме перечисленных; `1` — что блок видим только на перечисленных страницах; `2` — что для определения видимости Drupal должен выполнить специальный PHP-код, заданный администратором.

- pages. Содержимое этого поля зависит от значения поля visibility. Если это значение равно 0 или 1, то это поле содержит список путей Drupal. Если значение visibility равно 2, то поле pages содержит специальный PHP-код, который необходимо выполнить, чтобы определить, нужно ли отображать блок.
- title. Пользовательский заголовок для блока. Если данное поле пусто, используется стандартный заголовок блока (из модуля, предоставляющего этот блок). Если оно содержит <none>, то заголовок для блока выводиться не будет. Иначе текст из данного поля будет использоваться в качестве заголовка блока.
- cache. Определяет способ кэширования данного блока. Значение -1 означает, что блок не должен кэшироваться; значение 1 — кэширование блока для каждой роли (значение по умолчанию в Drupal для блоков, в которых не указан параметр кэширования); значение 2 — кэширование блока для каждого пользователя; значение 4 — кэширование для каждой страницы. Значение 8 означает, что блок будет кэшироваться одинаково для всех ролей, пользователей и страниц.

Использование блочных хуков

Блочные хуки — hook_block_info(), hook_block_configure(), hook_block_save() и hook_block_view() — управляют всей логикой программного создания блоков. Использование этих хуков позволяет объявить один блок или целый набор блоков. Функцию hook_block() для создания блоков можно реализовать в любом модуле. Рассмотрим все эти хуки.

- hook_block_info() — определяет все блоки, предоставляемые модулем.
- hook_block_configure($delta = '') — форма настройки для блока. Параметр $delta представляет собой идентификатор возвращаемого блока и может быть целочисленным или строковым. Этот же параметр используется в хуках hook_block_save и hook_block_view.
- hook_block_save($delta = '', $edit = array()) — сохраняет параметры конфигурации для блока. Параметр $edit содержит данные, отправленные из формы настройки для блока.
- hook_block_view($delta = '') — обрабатывает блок при активации в области, чтобы вывести его содержимое.

Создание блоков

В приведенном ниже примере мы создадим два блока, предназначенные для облегчения модерирования контента. Вначале нужно создать блок для списка комментариев, которые ожидают утверждения, а затем создать блок для списка неопубликованных нод. Оба блока будут содержать ссылки на форму редактирования для каждого фрагмента модерируемого контента.

Для размещения кода блоков создадим новый модуль с именем approval.module. Создайте в каталоге sites/all/modules/custom новую папку с именем approval (если папки modules и custom не существуют, понадобится создать и их).

Теперь добавим в эту папку файл approval.info:

```
name = Approval
description = Blocks for facilitating pending content workflow.
package = Pro Drupal Development
core = 7.x
version = VERSION
files[] = approval.module
```

Затем добавим файл `approval.module`:

```
<?php

/**
 * @file
 * Реализация различных блоков для рабочего потока обработки ожидающего контента.
 */
```

После создания этих файлов активируйте модуль со страницы Modules (Модули). Нам еще предстоит поработать с файлом `approval.module`, поэтому не закрывайте текстовый редактор.

Добавим текст хука `block_hook_info`, чтобы блок появился в списке блоков на странице администрирования блоков (рис. 10.4). Определим заголовок блока с помощью атрибута `info`, состояние — `True` (автоматически включен), область — `sidebar_first`, вес равен `0`, а видимость — `1` (видим):

```
/**
 * Реализация hook_block_info().
 */
function approval_block_info() {

  $blocks['pending_comments'] = array(
    'info'          => t('Pending Comments'), // Ожидающие комментарии
    'status'        => TRUE,
    'region'        => 'sidebar_first',
    'weight'        => 0,
    'visibility'    => 1,
  );
  return $blocks;
}
```

Обратите внимание, что значение `info` — это не заголовок блока, который отображается пользователям, если блок активен; это описание, выводимое только в списке блоков, доступных администратору для настройки. А сам заголовок блока будет реализован позже. Но сначала нужно задать дополнительные параметры настройки. Для этого понадобится реализовать функцию `hook_block_configure`, приведенную в следующем кодовом фрагменте. Мы создадим новое поле формы, видимой после щелчка на ссылке configure (настройка) рядом с блоком на странице администрирования блоков (рис. 10.5).

Рис. 10.4. Теперь блок Pending comments (Ожидающие комментарии) находится на странице обзора блоков под заголовком области Sidebar First (Первая врезка)

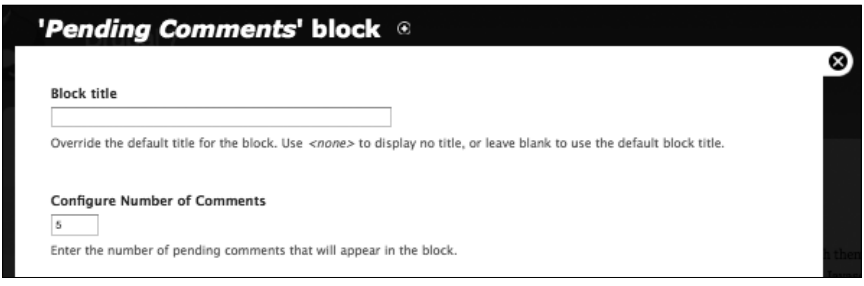

Рис. 10.5. Форма настройки блока со специальными полями

```
/**
 * Реализация hook_block_configure().
 */
function approval_block_configure($delta) {

  $form = array();

  switch($delta) {

  case 'pending_comments':
    $form['pending_comment_count'] = array(
      '#type' => 'textfield',
      '#title' => t('Configure Number of Comments to Display'),
                  // Укажите количество выводимых комментариев
      '#size' => 6,
      '#description' => t('Enter the number of pending comments that will
appear in the block.'),
                        // Введите количество ожидающих комментариев,
                        // которые будут отображаться в блоке.
      '#default_value' => variable_get('pending_comment_count',  5),
    );
    break;

  }

  return $form;

}
```

После отправки формы настройки блока, показанной на рис. 10.5, должен сработать хук `hook_block_save()`. Его можно использовать для сохранения значения поля формы:

```
/**
 * Реализация hook_block_save().
 */
function approval_block_save($delta = '', $edit = array()) {
  switch($delta) {
    case 'pending_comments':
     variable_set('pending_comment_count', (int)$edit['pending_comment_count']);
     break;
  }
  return;
}
```

Количество ожидающих комментариев, которое следует выводить, сохраняется с помощью встроенной системы переменных Drupal, а именно — с помощью функции `variable_set()`. Обратите внимание на приведение значения к целому виду в качестве проверки чистоты данных.

И, наконец, добавим операцию просмотра с помощью хука `hook_block_view` и пользовательской функции, которая возвращает список ожидающих комментариев при отображении блока:

```
/**
 * Реализация hook_block_view().
 */
function approval_block_view($delta = '') {

  switch ($delta) {
    case 'pending_comments':
      $block['subject'] = t('Pending Comments'); // Ожидающие комментарии
      $block['content'] = approval_block_contents($delta);
      return $block;
      break;
  }
}

/**
 * Функция отображения контента блока, определенная в модуле.
 */
function approval_block_contents($delta) {
  switch ($delta) {
    case 'pending_comments':
      if (user_access('administer comments')) {
         $nbr_comments = variable_get('pending_comment_count');
         $result = db_query("SELECT cid, subject FROM {comment} WHERE status = 0
           limit $nbr_comments");
         $items = array();
         foreach ($result as $row) {
           $items[] = l($row->subject, 'comment/' . $row->cid . '/edit');
         }
         return array('#markup' => theme('item_list', array('items' => $items)));
      }
    break;
  }
}
```

Рис. 10.6. Блок со списком ожидающих комментариев после его активации; содержит три ожидающих комментария

Здесь выполняется запрос из базы данных комментариев, которые ожидают утверждения, и вывод заголовков этих комментариев в качестве ссылок на сами комментарии, как показано на рис. 10.6.

Заголовок блока задается следующей строкой:

```
$block['subject'] = t('Pending comments');
                    // Ожидающие комментарии
```

Теперь блок вывода ожидающих комментариев завершен, и в этой функции `approval_block()` можно определить другой блок — для вывода всех неопубликованных нод со ссылками на страницу их редактирования:

```
/**
 * Реализация hook_block_info().
 */
function approval_block_info() {
  $blocks['pending_comments'] = array(
    'info'          => t('Pending comments'), // Ожидающие комментарии
    'status'        => TRUE,
    'region'        => 'sidebar_first',
    'weight'        => 0,
    'visibility'    => 1,
  );
  $blocks['unpublished_nodes'] = array(
    'info'          => t('Unpublished nodes'), // Неопубликованные ноды
    'status'        => TRUE,
    'region'        => 'sidebar_first',
    'weight'        => 0,
    'visibility'    => 1,
  );
  return $blocks;
}
```

Обратите внимание на указание ключей для блоков: `$blocks['pending_comments']`, `$blocks['unpublished_nodes']`, ..., `$blocks['xxxxxx']`. Модуль блоков будет использовать эти ключи по порядку в качестве параметра `$delta`.

Добавим к функциям `hook_block_configure` и `hook_block_save` форму для установки количества отображаемых нод и сохранения значения, введенного в форме администратором сайта:

```
/**
 * Реализация hook_block_configure().
 */
function approval_block_configure($delta) {
  $form = array();

  switch($delta) {

  case 'pending_comments':
    $form['pending_comment_count'] = array(
      '#type' => 'textfield',
      '#title' => t('Configure number of comments to display'),
                  // Укажите количество выводимых комментариев
      '#size' => 6,
      '#description' => t('Enter the number of pending comments that will
appear in the block.'),
                        // Введите количество ожидающих комментариев,
                        // которые будут отображаться в блоке.
      '#default_value' => variable_get('pending_comment_count',  5),
    );
    break;

  case 'unpublished_nodes':
    $form['unpublished_node_count'] = array(
      '#type' => 'textfield',
      '#title' => t('Configure Number of Nodes to Display'),
                  // Укажите количество выводимых нод
      '#size' => 6,
```

```
      '#description' => t('Enter the number of unpublished nodes that will
appear in the block.'),
                        // Введите количество неопубликованных нод,
                        // которые будут отображаться в блоке.
      '#default_value' => variable_get('unpublished_node_count',  5),
    );
    break;
  }
  return $form;
}
/**
 * Реализация hook_block_save().
 */
function approval_block_save($delta = '', $edit = array()) {
  switch($delta) {
    case 'pending_comments':
      variable_set('pending_comment_count', (int)$edit['pending_comment_count']);
      break;
    case 'unpublished_nodes':
      variable_set('unpublished_nodes_count', (int)$edit['unpublished_node_count']);
      break;
  }
  return;
}
```

После этого добавим в функции `hook_block_view` и `approval_block_content` возможность вывода неопубликованных нод:

```
/**
 * Реализация hook_block_view().
 */
function approval_block_view($delta = '') {
  switch ($delta) {
    case 'pending_comments':
      $block['subject'] = t('Pending Comments'); // Ожидающие комментарии
      $block['content'] = approval_block_contents($delta);
      return $block;
      break;
   case 'unpublished_nodes':
      $block['subject'] = t('Unpublished Nodes'); // Неопубликованные ноды
      $block['content'] = approval_block_contents($delta);
      return $block;
      break;
  }
}
/**
 * Функция отображения контента блока, определенная в модуле.
 */
function approval_block_contents($delta) {
  switch ($delta) {
    case 'pending_comments':
      if (user_access('administer comments')) {
        $nbr_comments = variable_get('pending_comment_count');
        $result = db_query("SELECT cid, subject FROM {comment} WHERE status = 0
          limit $nbr_comments");
        $items = array();
```

```
        foreach ($result as $row) {
          $items[] = l($row->subject, 'comment/'.$row->cid.'/edit');
        }
        return array('#markup' => theme('item_list', array('items' => $items)));
      }
    break;

    case 'unpublished_nodes':
      if (user_access('administer nodes')) {
        $nbr_nodes = variable_get('unpublished_node_count');
        $result = db_query("SELECT nid, title FROM {node} WHERE status = 0
          limit $nbr_nodes");
        $items = array();
        foreach ($result as $row) {
          $items[] = l($row->title, 'node/'.$row->nid.'/edit');
        }
        return array('#markup' => theme('item_list', array('items' => $items)));
      }
    break;
  }
}
```

Результат работы только что созданного блока для отображения неопубликованных нод показан на рис. 10.7.

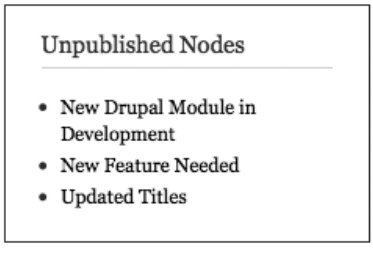

Рис. 10.7. Блок со списком неопубликованных нод

Активация блока при инсталляции модуля

В модуле `approval.module` мы автоматически активировали блоки и назначали им область темы. Например, при создании блока `Pending Comments` он был автоматически активирован (`status = TRUE`) и назначен области `sidebar_first`:

```
$blocks['pending_comments'] = array(
    'info'          => t('Pending Comments'), // Ожидающие комментарии
    'status'        => TRUE,
    'region'        => 'sidebar_first',
    'weight'        => 0,
  );
```

В некоторых ситуациях требуется разрешить администратору сайта определять, активировать ли блок и какой области темы его назначить. В таких случаях понадобится установить атрибут `status` в `FALSE` и не назначать блоку никакой области. В следующем примере показано создание нового блока `Pending Users`, который не активируется автоматически, и которому не назначается область:

```
$blocks['pending_users'] = array(
    'info'          => t('Pending Users'), // Ожидающие пользователи
    'status'        => FALSE,
    'weight'        => 0,
  );
```

Примеры видимости блоков

В интерфейсе администрирования блоков имеется раздел Page visibility settings (Параметры видимости страницы), где можно ввести фрагменты PHP-кода. При построении страницы Drupal выполняет такие фрагменты, чтобы определить, нужно ли выводить блок. Ниже приведены примеры наиболее ходовых фрагментов, каждый из которых возвращает значение `TRUE` или `FALSE`, означающее, что блок должен быть видимым или невидимым при данном запросе.

Отображение блока только для вошедших пользователей

Возвращает `TRUE`, только если значение `$user->uid` не равно `0`:

```
<?php
  global $user;
  return (bool) $user->uid;
?>
```

Вывод блока только для анонимных пользователей

Возвращает `TRUE`, только если значение `$user->uid` равно `0`:

```
<?php
  global $user;
  return !(bool) $user->uid;
?>
```

Резюме

После прочтения данной главы вы должны уметь выполнять следующие действия.

- Знать, что собой представляют блоки и чем они отличаются от нод.
- Знать, как работают параметры видимости и размещения блоков.
- Знать, как определить один или несколько блоков.
- Знать, как сделать блок активным по умолчанию.