# Drupal: разработка модуля

Роман Архаров

Данная статья — продолжение материала, посвященного CMS Drupal (см. PC Magazine/RE, 12/2008). В первой статье подробно рассказано о назначении и возможностях системы, а также приведены примеры сборки сайтов на Drupal с использованием уже существующих модулей. Этот же материал будет больше интересен техническим специалистам, умеющим программировать на языке PHP, знакомым с основами HTML и CSS, и тем, кто хочет больше узнать о методах разработки собственных модулей для этой системы. Предыдущая статья доступна сейчас в Интернете по адресу: *www.pcmag.ru/solutions/detail.php?ID=32535*. Перед чтением этого материала рекомендуется освежить в памяти информацию, просмотрев ее первые три раздела.

## Разработка собственного модуля

Система управления сайтом Drupal построена по модульному принципу: компактный набор служебных функций (ядро) расширяется при помощи модулей — файлов с PHP-кодом. Модули должны содержать «хуки» (hooks) — особым образом именованные функции, которые вызываются ядром Drupal при возникновении каких-либо событий. Каждый модуль имеет системное имя, которое должно состоять из латинских букв, цифр, знака подчеркивания (и начинаться обязательно с буквы). Имя хука должно состоять из двух частей: имени модуля и названия события. При возникновении любого события ядро Drupal в каждом из установленных модулей ищет и выполняет соответствующую функцию, т. е. функцию с именем название_модуля_название_события. Например, при возникновении событий, связанных с учетной записью пользователя (регистрация, авторизация, изменение роли пользователя и др.), ядро Drupal вызывает функции, реализующие хук hook_user, поэтому, чтобы модуль с именем example мог отреагировать на это событие, в нем необходимо объявить функцию с именем example_user(). Список передаваемых в эту функцию аргументов, пример ее использования и информацию обо всех функциях и хуках, доступных в Drupal, можно найти на странице официальной документации *http://api.drupal.org* или ее русской версии: *http://api.drupal.ru*.

Каждый модуль для Drupal представляет два файла или более, которые должны находиться в папке sites/all/modules/*название_модуля**.

* Строго говоря, модули могут находиться не только в sites/all/modules, но и в некоторых других папках, например sites/example.com/modules или sites/all/modules/example/example_submodule, но описание таких конфигураций выходит за рамки этой статьи.

В файле *название_модуля*.info должна находиться служебная информация, а в файле *название_модуля*.module — исходный текст. При наличии этих двух файлов модуль станет доступным на странице установки модулей Drupal (Administer — Modules, admin/build/modules). Кроме того, в этой же папке может находиться необязательный файл *название_модуля*.install, содержащий реализации хуков, которые будут выполнены при инсталляции модуля. В этом файле обычно располагаются инструкции, создающие новые таблицы в базе данных и задающие значения по умолчанию для настроек модуля.

Для иллюстрации использования системы хуков я приведу пример разработки простого модуля, который формирует блок с информацией о курсах валют. Этот модуль при выполнении соответствующей строки cron-таблицы будет соединяться с сервером ЦБ РФ и получать от него информацию о курсах валют. На основе полученных данных и настроек, заданных через интерфейс управления модулем, а также функций темизации будет генерироваться выходный HTML-код. Он будет кэшироваться стандартными средствами Drupal (благодаря чему администратор при желании сможет перенести этот кэш из базы данных, например, в файловую систему или memcache) и выдаваться по запросу пользователя в виде блока.

Модуль получит название currencies. На первом этапе его разработки необходимо (относительно корня Drupal-сайта) создать папку sites/all/modules/currencies, в которой мы будем сохранять новые файлы.

### currencies.info

В .info-файлах модулей содержится служебная информация, без которой модуль не будет виден в системе. Начинаться любой .info-файл должен со строки

```
; $Id$
```

В файлах с PHP-кодом после открывающего тега <?php необходимо добавить строку

```
// $Id$
```

Эту строку, если модуль будет размещен в официальном CVS-репозитории Drupal, заменит служебная информация.

Далее в файле .info должны располагаться три обязательных параметра: название модуля, его описание и версия ядра Drupal, с которой работает модуль. Кроме того, в этом файле могут находиться необязательные параметры: минимальная версия PHP, необходимая для запуска модуля, зависимость от других модулей Drupal, без которых текущий модуль не будет работать, и пр. Подробное

описание всех доступных к использованию в .info-файле параметров можно найти в официальной документации (ссылка на эту и другие цитируемые в статье страницы документации размещена во врезке «Ссылки на документацию»).

В нашем случае файл currencies.info будет иметь такой вид:

```
; $Id$
name = Currencies block
description = Show currencies
core = 6.x
```

Хотя один файл в нашем модуле уже есть, но пока он отсутствует в списке имеющихся в системе модулей, и мы переходим к следующему файлу.

**currencies.install**

По стандартам кодирования Drupal перед каждой функцией, реализующей хук, должен быть размещен комментарий вида:

```
/*
 * Implementation of hook_название_хука().
 */
```

Аналогичный комментарий (только без текста Implementation of...) с описанием функции и ее аргументов должен располагаться перед любой другой функцией. Эти комментарии используются системой генерации документации Doxygen,

*Система управления сайтом Drupal построена по модульному принципу: ядро расширяется при помощи модулей — файлов с PHP-кодом.*

по ним также удобно вести поиск. Здесь для экономии места такие комментарии опущены.

При инсталляции и деинсталляции модулей вызываются хуки hook_install и hook_uninstall. Отмечу, что в Drupal кроме понятий инсталляции и деинсталляции есть понятия активации и деактивации модуля. Если модуль устанавливается впервые (в административном интерфейсе, в списке модулей установлена галочка напротив нужного модуля и нажата кнопка Submit), сначала происходит событие install, затем событие enable, т. е. ядро Drupal ищет и, если находит, вызывает функции, реализующие хуки hook_install и hook_enable для устанавливаемого модуля. Далее, если администратор выключает модуль, то происходит событие disable и вызывается функция, реализующая хук hook_disable. В следующий раз, когда модуль будет вновь включен, произойдет только событие enable, а не install. Если модуль был сначала деактивирован, а затем удален (удаление производится на отдельной от списка модулей странице), то происходят события disable и uninstall и в следующий раз при включении модуля опять произойдут события install и enable.

Такое разделение очень удобно. Обычно при возникновении события install программисты создают необходимые для работы модуля таблицы в базе данных, а при событии uninstall — удаляют их, таким образом после деинсталляции модуля в системе не остается никаких свидетельств его присутствия. При включении и выключении модуля (enable и disable) никакие сохраненные

*При возникновении события install программисты создают необходимые для работы модуля таблицы в базе данных*

модулем данные не удаляются, а лишь отключается функционал модуля.

Вернемся к нашему примеру. Информацию о курсах валют разрабатываемый модуль currencies будет получать с сервера Центрального Банка РФ. Чтобы при каждом показе блока не обращаться с запросом к удаленному серверу, данные будут сохраняться в базе данных нашего сайта. В самом простом случае для хранения данных можно было бы воспользоваться функциями cache_set или variable_set из ядра Drupal, однако такой подход не очень удобен, когда нужно хранить информацию о курсах валют за длительный период, например, для последующего ее анализа. Поэтому мы создадим отдельную таблицу в базе данных и в ней будем хранить всю полученную информацию.

Наш пример довольно прост, поэтому события enable и disable использоваться в нем не будут, а на install и uninstall мы назначим функции создания и удаления таблицы в БД, для чего в файле currencies.install разместим функции, реализующие хуки hook_install и hook_uninstall:

```
function currencies_install() {
  drupal_install_schema('currencies');
}

function currencies_uninstall() {
  drupal_uninstall_schema('currencies');
}
```

Функции drupal_install_schema и drupal_uninstall_schema являются частью Drupal Schema API. Schema API — это слой абстракции от базы данных, благодаря которому программист может не задумываться о том, как адаптировать свой SQL-запрос под ту или иную базу данных, ему достаточно сформировать массив определенного вида и передать его одной из функций Schema API, после чего этот массив будет преобразован в корректный SQL-запрос к той базе данных, которая используется с Dupal. Единственный аргумент, который принимают эти две функции, — название

модуля, схема которого будет установлена или удалена, т. е. после вызова drupal_install_schema('currencies') Drupal попытается создать таблицы, описанные в реализации hook_schema модуля currencies, поэтому нужно создать эту реализацию (см. листинг 1). Этот хук должен возвращать ассоциативный массив, содержащий информацию о создаваемых таблицах. В нашем примере создается таблица с именем currencies_block и двумя полями: timestamp и data. Подробное описание формата возвращаемого массива можно найти в документации.

На данном этапе в нашем модуле currencies есть два файла: currencies.info и currencies.install, но он по-прежнему недоступен для выбора на странице со списком модулей, поскольку в нем отсутствует самый важный файл — .module. Если в папке модуля создать файл currencies.module и разместить в нем всего две строчки:

```
<?php
// ; $Id$
```

(их описание было дано выше), модуль тут же станет доступным для установки, однако, так как файл currencies.module не содержит никаких инструкций, установка этого модуля приведет только к созданию одной таблицы в БД и он не будет нам полезен. Поэтому мы переходим к самому большому, сложному и важному этапу — разработке основного функционала модуля.

### currencies.module

В Drupal реализован механизм «ролей» и «прав доступа». По умолчанию в системе есть две «роли» (группы) пользователей — анонимы и авторизованные пользователи, одна из которых автоматически присваивается каждому пользователю в зависимости от того, авторизован он в системе или нет. Каждый модуль может объявить «права доступа» к своим ресурсам, после чего администратор сайта через специальный интерфейс может установить, какие роли имеют доступ к каким сервисам сайта. Например, обычно пользователи, входящие в группу авторизованных, могут вести персональный блог, а у анонимов такой возможности нет. В любой момент администратор сайта через Web-интерфейс может создать дополнительные роли (модераторы, администраторы и т. п.), определить особые разрешения для каждой из ролей (например, модераторы могут снимать с публикации материалы и комментарии, а администраторы их безвозвратно удалять) и назначать эти роли любому пользователю, причем ему одновременно может быть присвоено несколько ролей.

*В интерфейсе администратора с помощью hook_menu и Forms API будет создана форма, позволяющая указать список валют.*

**Листинг 1**

```
function currencies_schema() {
  $schema['currencies_block'] = array(
    'description' => t('Some table description.'),
    'fields' => array(
      'timestamp' => array(
        'type' => 'int',
        'size' => 'normal',
        'not null' => TRUE,
        'default' => 0,
      ),
      'data' => array(
        'type' => 'text',
        'not null' => TRUE,
      ),
    ),
  );
  return $schema;
}
```

Выше уже была описана логика работы разрабатываемого модуля, сейчас, перед тем как написать первые строки кода, необходимо более подробно продумать алгоритм работы программы. Если говорить о нашем случае, то программа будет разбита на четыре основные части:

- интерфейс администратора;
- автоматически выполняемые процедуры;
- функции темизации;
- интерфейс пользователя.

В интерфейсе администратора с помощью hook_menu и Forms API будет создана форма, позволяющая администратору указать список валют, которые необходимо выводить в блоке, адрес XML-документа, из которого будет «подтягиваться» информация об обновленных курсах валют, и частоту обновления данных. Кроме того, с помощью hook_perm будет создано «право доступа», дающее возможность администратору сайта ограничить доступ к настройкам модуля.

Регулярно выполняемая процедура в модуле будет одна: при запуске строки cron-таблицы необходимо проверить, когда произошло последнее обновление данных о курсах валют. Если между текущим моментом и последним обновлением данных прошло время большее, чем указано в соответствующей настройке интерфейса управления модулем, то необходимо соединиться с сервером ЦБ, получить обновленные данные и сохранить их в базе.

Интерфейс пользователя создадим при помощи хука hook_block, позволяющего формировать блоки с данными. Чтобы верстальщик мог изменять внешний вид выводимых модулем данных, создадим свою функцию темизации, которую необходимо зарегистрировать в системе при помощи hook_theme.

## Административный интерфейс

Теперь приступим к реализации каждого из описанных этапов. Для начала добавим в наш файл currencies.module функцию:

```
function currencies_perm() {
  return array('access currencies block settings');
}
```

Эта функция — реализация хука hook_perm, который, как сказано выше, определяет дополнительные права доступа. Hook_perm — один из самых простых хуков, он всего лишь возвращает массив строк, представляющих собой права доступа. После инсталляции модуля администратор сайта на странице Admin — Permissions (admin/user/permissions) может указать, какие группы пользователей имеют право доступа access currencies block settings, а мы в дальнейшем, во время реализации формы настроек модуля при помощи функции user_access, будем проверять, имеет ли текущий пользователь право доступа access currencies block settings или нет.

Важный момент. Пользователь с uid=1, т. е. первый созданный в системе, является суперпользователем, для него функция user_access всегда возвращает значение TRUE, а это значит, что он всегда имеет доступ ко всем функциям сайта. Это одна из причин, почему не рекомендуется работать в системе с учетной записью суперпользователя: зачастую разработчики забывают раздавать пользователям необходимые права доступа, так как сами, работая как суперпользователи, не имеют проблем с доступом к ресурсам сайта.

Для определения дополнительного системного пути, по которому в нашем примере будет доступна страница управления модулем, необходимо создать реализацию хука hook_menu:

```
function currencies_menu() {
  $items = array();
  $items['admin/settings/cur-block'] = array(
    'title' => t('Currencies block settings'),
    'description' => 'Currencies block settings.',
    'page callback' => 'drupal_get_form',
    'page arguments' => array('currencies_settings'),
    'access arguments' => array('access cur block settings'),
  );
  return $items;
}
```

Эта функция также возвращает ассоциативный массив. Ключом каждого элемента массива должен быть путь, регистрируемый в системе (в нашем случае это admin/settings/cur-block), а значением — вложенный массив, содержащий информацию о создаваемом пункте меню. Давайте разберем каждый из параметров отдельно.

Title — заголовок меню — будет использоваться при переходе на страницу с адресом admin/settings/cur-block в строке заголовка браузера (тег <title>) и в качестве заголовка страницы (тег <h1>), а также в качестве текста ссылки, ведущей на созданную страницу настроек.

Description — описание пункта меню, которое в нашем случае будет использоваться на странице администрирования.

Page callback — функция, которая будет генерировать страницу, создаваемую по указанному пути. В простом случае значением этого параметра должна быть функция, возвращающая HTML-код, который будет показан пользователю. Однако мы по указанному адресу создаем не обычную страницу, а форму, значения которой автоматически сохраняются в БД. Поэтому для параметра page callback мы назначаем вызов системной функции drupal_get_form(), которая выведет на экран форму, созданную функцией с именем, указанным в элементе массива page arguments; в нашем случае это функция currencies_settings(). Функция currencies_settings() должна вернуть ассоциативный массив, содержащий информацию об элементах создаваемой формы. Подробнее об этом массиве будет рассказано ниже.

Access arguments — массив «прав доступа». Пользователи, обладающие правами доступа, перечисленными в этом массиве, могут получить доступ к создаваемому пункту меню.

### Листинг 2

```
function currencies_settings() {
  $form['currencies_list'] = array(
    '#type' => 'textfield',
    '#title' => t('Currencies'),
    '#default_value' => variable_get('currencies_list',
      "USD,EUR,CNY,BYR,KZT,TRY,UAH,JPY"),
    '#maxlength' => 255,
  );

  $form['currencies_list_freq'] = array(
    '#type' => 'textfield',
    '#title' => t('Frequency of updating of the data
      (in seconds)'),
    '#default_value' => variable_get('currencies_list_freq',
      3600),
    '#maxlength' => 255,
    '#description' => t('It is recommended to use value
      not less than 3600.'),
  );

  $form['currencies_list_url'] = array(
    '#type' => 'textfield',
    '#title' => t('Адрес xml-файла'),
    '#default_value' => variable_get('currencies_list_url',
      "http://www.cbr.ru/scripts/XML_daily.asp?date_req=
      %d/%m/%y"),
    '#maxlength' => 255,
    '#description' => t('The XML-file address.'),
  );
  return system_settings_form($form);
}
```

## Подготовка Web-страницы

Тема оформления в Drupal — это набор особым образом сформированных HTML-шаблонов и CSS-файлов, на основе которых ядро Drupal генерирует запрашиваемую пользователем страницу. Если в системе используется встроенный в Drupal шаблонный «движок» PHPTemplate, то каждая тема оформления может содержать служебный файл template.php, в котором могут размещаться функции, переопределяющие стандартный вывод модулей. У каждой темы оформления, как и у каждого модуля, должно быть свое уникальное имя и файл настроек .info.

Более подробную информацию о параметрах пунктов меню можно найти в документации.

Сейчас в нашем модуле определен новый пункт меню, но не определена функция, формирующая содержимое страницы, на которую этот пункт указывает (см. листинг 2).

Как и хуки hook_menu, hook_schema и многие другие хуки Drupal, эта функция должна возвращать ассоциативный массив, на этот раз содержащий информацию о параметрах создаваемой формы. Здесь мы создаем три однострочных текстовых поля (параметр #type имеет значение textfield), значения по умолчанию для которых (параметр #default_value) будут храниться и выбираться из стандартной таблицы variables Drupal при помощи функций variable_set() и variable_get(). Благодаря использованию функций drupal_get_form и system_settings_form нет необходимости заботиться о создании кнопок Submit и Reset, а также о функциях, обрабатывающих и сохраняющих данные формы. В более сложных случаях, которые будут рассмотрены в следующей статье, придется вручную создавать функции проверки введенных пользователем значений и сохранения данных.

Подробное описание типов полей, используемых в формах, можно найти в документации.

Все, мы завершили разработку первой из трех частей нашего модуля — административного интерфейса и переходим к разработке второй его части — инструмента получения данных от удаленного сервера.

### Регулярные процедуры

Чтобы Drupal периодически выполнял определенные действия, в планировщике задач операционной системы необходимо настроить запуск файла cron.php, который находится в корне каждого Drupal-сайта. При выполнении этого файла будет вызываться хук hook_cron, и в нашем модуле мы напишем его реализацию:

```
function currencies_cron() {
  currencies_contents();
}
```

Процедура получения и обработки XML-файла, расположенного на удаленном сервере, довольно обычна, поэтому она здесь не приведена. При желании вы можете самостоятельно разобрать логику работы этой функции, изучив исходные коды модуля Currencies, которые есть на диске, прилагаемом к журналу. В результате ее работы формируется массив $result, в котором содержится информация о курсах валют и который передается функции темизации. Функцию currencies_contents() см. в листинге 3. Здесь мы сначала проверяем, нет ли запрашиваемых данных в кэше Drupal, если нет, то происходит соединение с удаленным сервером (его адрес указан через административный интерфейс модуля), получение и обработка XML-файла и формирование массива данных, который передает функции темизации. Функция темизации формирует выходной HTML-код, записываемый в кэш, его время жизни явно указывается исходя из соответствующей настройки, заданной в интерфейсе управления модулем. Drupal автоматически управляет закэшированными данными и при необходимости удаляет устаревшие записи.

Функция currencies_contents() будет использоваться не только при запуске cron-задания, но и при формировании блока с данными, который будет показываться пользователю. Таким образом, практически всегда пользователи будут видеть данные, полученные из кэша Drupal, если же на сайте не работает cron и нет закэшированных данных о курсах валют, то произойдет соединение с сервером ЦБ, формирование и запись в кэш необходимых данных.

### Функции темизации

Теперь подробнее остановимся на функциях темизации (theming; термин не слишком благозвучен, но уже стал общепринятым). Теоретически прямо в коде функции currencies_contents() можно было бы сформировать HTML-код, который в дальнейшем и видел бы посетитель сайта в браузере. Однако такой подход неверен, поскольку при изменении оформления данных пришлось бы менять код модуля, а это влечет за собой две проблемы. Во-первых, модуль могут использовать сторонние разработчики, и им для внесения изменений придется вникнуть в структуру модуля и внести в него изменения, которые могут привести к ошибкам. Во-вторых, часто при разработке крупных проектов версткой и программированием занимаются разные люди. Верстальщик может не иметь представления

## Листинг 3

```
function currencies_contents() {
  if(!$c = cache_get('currencies')) {
    /* здесь пропущен код, отвечающий за получение
       и обработку XML-файла */
    $output = theme('currencies_block', $result);
    $t = variable_get('currencies_list_freq', 3600);
    if(!$t || !is_numeric($t)) $t = 3600;
    cache_set("currencies", $output, 'cache', time() + $t);
  } else {
    $output = $c->data;
  }
  return $output;
}
```

о том, как работать с языком PHP в целом и модулями Drupal в частности. По этому правильнее вынести все действия, связанные с оформлением данных, в отдельные файлы и функции, для чего и нужны функции темизации.

Функции темизации — это функции, генерирующие HTML-код, который впоследствии показывается пользователю. Особенность таких функций в том, что, во-первых, они не должны реализовывать никакой бизнес-логики, т. е. в их задачи входит только генерирование HTML-кода на основе полученных аргументов. Во-вторых, эти функции могут быть переопределены разработчиком сайта без редактирования кода модуля, путем изменения файла template.php или создания файла-шаблона.

В нашем случае массив данных формируется функцией currencies_contents(), а HTML-код создается функцией

*Функции темизации — это функции, генерирующие HTML-код, который впоследствии показывается пользователю.*

theme_currencies_block(), которую нам сейчас предстоит определить и зарегистрировать. Подчеркну, что вызов любой функции темизации осуществляется через вызов функции-обертки с именем theme(), т. е. функция theme_currencies_block($argument) должна вызываться как theme('currencies_block', $argument). Чем вызвана эта необходимость, я объясню чуть позже.

Для регистрации функций темизации, используемых в модуле, мы должны реализовать hook_theme, который возвращает массив имен применяемых функций темизации и принимаемых ими параметров. В нашем модуле будет использоваться одна функция темизации theme_currencies_block(), которая на вход принимает один обязательный аргумент — массив курсов валют, поэтому реализация хука hook_theme будет выглядеть так:

```
function currencies_theme() {
  return array(
    'currencies_block' => array(
      'arguments' => array('result' => NULL),
    ),
  );
}
```

Без такой регистрации вызов theme('currencies_block', $argument) будет невозможен. Сама функция темизации будет такой:

```
function theme_currencies_block($result) {
  $output = "<ul>";
  foreach($result as $k => $v) {
    if($v["diff"] > 0) {
      $color = "#5aaf43";
      $v["diff"] = "+" . $v["diff"];
    }
    else if($v["diff"] < 0) $color = "#f00";
    else if($v["diff"] == 0) $color = "#00f";
    $output .= "<li>" . $v["nominal"] . " " . $v["name"] . " = "
      . $v["value"] . " (<span style=\"color: " . $color . "\">"
      . $v["diff"] . "</span>)</li>";
  }
  $output .= "</ul>";
  return $output;
}
```

Еще раз обратите внимание на то, что в функции currencies_theme в качестве имени функции темизации указывается название currencies_block, а реализация ее имеет имя theme_currencies_block(). Нужно это, чтобы в дальнейшем разработчики сайтов, использующие наш модуль, могли переопределить эту функцию, т. е. изменить формируемый ею HTML-код. Для этого им в папке со своей темой в файле template.php достаточно будет создать функцию с именем, совпадающим с именем функции темизации, но в котором слово theme заменено на название используемой темы оформления, т. е. создать функцию *название_темы_оформления*_currencies_block(). Ядро Drupal, когда встретит в коде модуля вызов вида theme('currencies_block', $result), сначала попробует найти функцию темизации в файле template.php используемой в данный момент темы оформления, и только если там ее не найдет, использует функцию theme_currencies_block().

В случае больших шаблонов удобнее вынести функцию темизации из файла template.php в отдельный файл. Для этого в массиве, возвращаемом реализацией хука hook_theme,

*Ядро Drupal пытается найти функцию темизации в файле template.php используемой в данный момент темы оформления.*

нужно добавить элемент с ключом template и именем, соответствующим имени файла-шаблона, а в каталоге с модулем разместить файл-шаблон с указанным ранее именем и расширением .tpl.php. В итоге hook_theme() примет вид:

```
function currencies_theme() {
  return array(
    'currencies_block' => array(
      'arguments' => array('result' => NULL),
      'template' => 'cur-block',
    ),
  );
}
```

а в папке с модулем нужно разместить файл с именем cur-block.tpl.php и таким содержимым:

```
<ul>
<?php
  foreach($result as $k => $v) {
    if($v["diff"] > 0) {
      $color = "#5aaf43";
      $v["diff"] = "+" . $v["diff"];
    }
    else if($v["diff"] < 0) $color = "#f00";
    else $color = "#00f";
    print "<li>" . $v["nominal"] . " " . $v["name"] . " = " .
      $v["value"] . " (<span style=\"color: " . $color . "\">" .
      $v["diff"] . "</span>)</li>";
  }
?>
</ul>
```

Теперь, чтобы переопределить вывод данных, формируемых модулем Currencies, разработчику достаточно скопировать файл cur-block.tpl.php из папки с модулем в папку с используемой темой оформления и внести в него необходимые изменения.

## Интерфейс пользователя

Сейчас наш модуль имеет интерфейс администратора, умеет автоматически соединяться с удаленным сервером, получать необходимую информацию, формировать из нее HTML-код и сохранять его в кэше. Осталась самая простая для программиста и наиболее значимая для посетителя сайта часть — вывод данных на экран.

Для решения этой задачи мы могли бы через созданную ранее функцию currencies_menu() зарегистрировать еще один путь и в качестве параметра page callback указать функцию currencies_contents(), которая получила бы необходимые данные (из кэша или от удаленного сервера) и вывела бы их на экран через функцию темизации. Но мы хотим, чтобы курсы валют выводились не отдельной страницей, а в блоке (оформленный особым образом элемент, содержащий данные) на любой заданной администратором сайта странице. Для этого необходимо создать реализацию хука hook_block() (см. листинг 4).

### Листинг 4

```
function currencies_block($op = 'list', $delta = 0,
    $edit = array()) {
  switch ($op) {
    case 'list':
    $blocks[0] = array(
      'info' => t('Currencies block'),
    );
    return $blocks;
    case 'view':
    switch ($delta) {
      case 0:
      $block['subject'] = t('Currencies block');
      $block['content'] = currencies_contents();
      break;
    }
    return $block;
  }
}
```

### Ссылки на документацию

- Руководство по разработке модулей: *http://drupal.org/node/206754.*
- Параметры, используемые в .info-файлах: *http://drupal.org/node/231036.*
- Введение в Schema API: *http://drupal.org/node/146843.*
- Типы полей, используемых в Schema API: *http://drupal.org/node/159605.*
- Описание hook_menu(): *http://api.drupal.org/api/function/hook_menu/6.*
- Описание типов полей Forms API: *http://api.drupal.org/api/file/developer/topics/forms_api_reference.html/6.*
- Описание hook_block(): *http://api.drupal.org/api/function/hook_block/6.*

Параметр $op (сокращение от operation), содержит информацию о том, какая операция над блоком в данный момент выполняется. Значение list сообщает модулю, что информация о нем выводится в списке модулей, здесь мы должны передать ядру Drupal один обязательный параметр — заголовок модуля и ряд необязательных параметров, информацию о которых можно найти в документации.

Значение view аргумента $op означает, что пользователь просматривает страницу, на которой должен быть отображен наш блок, и модуль обязан вернуть его содержимое. Для этого функция должна сформировать и выдать ассоциативный массив, элемент с ключом content которого будет использован в качестве содержимого блока, а необязательный элемент с ключом subject — заголовка.

Каждый модуль может создавать несколько блоков, параметр $delta хука hook_block содержит индекс обрабатываемого в текущий момент блока.

Все, разработка модуля завершена! Теперь администратор сайта, после инсталляции модуля Currencies, может активировать созданный им блок, для этого необходимо перейти в меню Administer — Blocks (admin/build/block), перетащить мышкой строку с описанием блока в нужный регион (область страницы, которая может содержать один или несколько блоков) и сохранить изменения. В свойствах модуля доступны стандартные для Drupal средства управления блоком: список страниц, на которых блок должен показываться (или наоборот список страниц, на которых блок не должен показываться), список ролей пользователей, которые имеют доступ к содержимому блока, и другие. PC