Система управления контентом SantaFox™

Руководство по разработке модулей

Версия 1.0

SantaFox developer
Saint-Peterburg, 2008

Оглавление

Введение

Для кого это руководство

Данная книга предназначена для людей, решивших попробовать свои силы в разработке новых модулей для системы управления контентом SantaFox™. Сначала мы дадим краткое представление об общей архитектуре системы и её компонентах. Обсудим, что собой представляют модули, и какие функциональные возможности они могут реализовать. Как происходит процесс вызова модуля и встраивание модуля в SantaFox™, а потом и в сам сайт.

Как работает SantaFox™ (коротко)

При разработке SantaFox™ была поставлена задача максимально отделить архитектуру модулей от общей архитектуры системы с тем, что бы модуль был более самостоятельным, и не требовалось изучать или использовать непосредственный программный код CMS при разработке новых модулей. Помимо этого, необходимо было реализовать схему, при которой администратор сайта мог легко конструировать сайт, используя установленные модули и возможности SantaFox™, вообще не прибегая к непосредственному программированию, в каком либо виде.

Для того, что бы разрабатывать новые модули для SantaFox™ вам, возможно, захочется иметь четкое представление о том, как работает CMS. Сначала немного общих принципов, среди которых можно выделить ряд основных, а именно:

- использование шаблонов страниц для формирования конкретной страницы;
- использование меток в шаблонах, для их замещения динамическим контентом, управляемым средствами SantaFox™;
- использование модулей для реализации большинства функций сайта по формированию динамического контента;
- использование принципов наследования;

При формировании страницы сайта SantaFox™ производит следующие действия:

1. Определяет по URL запрашиваемую страницу
2. Определяет шаблон, используемый для данной страницы
3. Определяет метки, используемые в шаблоне
4. Определяет, какие модули и действия привязаны к каждой из меток.
5. Вызов необходимых модулей и их действий
6. Замещение меток, тем контентом, который сформирован привязанным действием метода
7. Вывод итоговой страницы

Метки и языковые переменные

Одним из основных понятий системы является понятие *метки*. Меткой называется текст вида [#<*имя метки*>#] который может быть размещен в любом месте шаблона. Метка определяет то место в шаблоне, куда будет выводиться тот или иной (полученный от разных компонент системы) динамический контент. В качестве *имени метки* могут быть использованы только буквы латинского алфавита, цифры и знак нижнего подчеркивания.

Идентичный синтаксис имеют и *языковые переменные,* используемые в административном интерфейсе. При разработке административной части любого модуля старайтесь использовать языковые переменные, с тем, что бы модуль можно было легко адаптировать под разные языки.
Значения языковых переменных для каждого языка описывается в отдельном файле для каждого языка.

Данные файлы должны размещаться в <*папка модуля*>/lang/<*2-у буквенный код языка*>.php
У каждого модуля может быть собственный файл с языковыми переменными, которые будут инсталлироваться вместе с модулем.

```
01.     <?php
02.     $type_langauge = '<2-у буквенный код языка>';
03.     $il['<ID языковой переменной>'] = '<представление>';
04.     $il['<ID языковой переменной>'] = '<представление>';
05.     …
06.     ?>
```

<*ID языковой переменной*> указывается без открывающих и закрывающих конструкций («[#» и «#]»), тогда как в шаблонах самих модулей необходимо указывать полный вид языковой переменной.

Что такое модуль и с чем его едят

Давайте поближе рассмотрим модуль, используемый SantaFox™. Определим какова его внешняя структура, с которой работает администратор сайта. Определим внутреннюю структуру модуля, которой должен придерживаться разработчик. Начнем с основополагающих понятий *базового модуля* и *дочернего модуля*.

Базовый модуль и его потомки

В CMS используется два понятия ***базовый модуль*** и ***дочерний модуль***.

Сразу оговорюсь, что это условные понятий с точки зрения разработчика, но между тем они достаточно важны. Данными понятиями оперируют администратор сайта, при подключении и настройки модуля. Однако разработчику необходимо знать, каким образом модуль может функционировать в системе, с тем, что бы более точно спроектировать модуль, для решения поставленных задач.

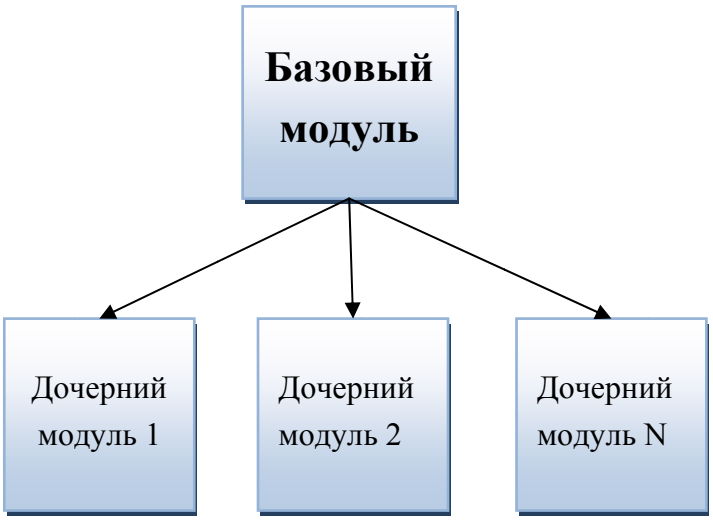

Рисунок 1: Взаимосвязи подчинённости базового модуля и дочерних модулей.

Дочерний модуль – это собственно сам модуль, который выполняет возложенные на него задачи по формированию динамического контента. Особенность дочерних модулей состоит в том, что их может быть произвольное количество, которое определяет администратор сайта исходя из возможностей модуля и стоящих перед ним (администратором) задач.

Базовый модуль – объект CMS, предназначенный для группировки дочерних модулей в один блок и возможности наследования дочерними модулями свойств от базового модуля.

Таким образом, базовый модуль является некоторой абстракцией, позволяющей объединить конкретные дочерние модули.

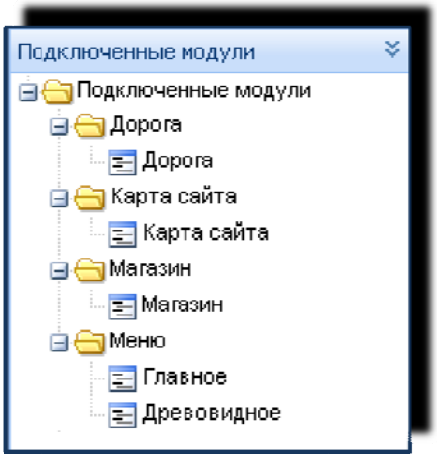

Рисунок 2: Пример подключённых модулей. Знаком "папки" отображается базовый модуль, у которого есть дочерние.

При разработке модуля, необходимо только четко понимать, что при работе модуля в системе происходит вызов именно дочерних модулей и их действий. Соответственно разработчик НЕ ЗНАЕТ и НЕ ДОЛЖЕН знать, какое количество дочерних модулей сейчас функционирует, а обращения ко всем свойствам модуля происходит через соответствующие функции ядра.

Помимо этого, понятия базового и дочернего модуля применяются при формировании административного интерфейса конкретного модуля. Административная часть модуля может:

- вызываться на каждый дочерний модуль;
- только на базовый;
- не вызываться вовсе.

Необходимость создавать более одного дочернего модуля вытекает из функциональных задач самого модуля, и потребность в этом определяется на этапе разработки модуля. Например, модуль формирующий дорогу сайта, будет иметь всего один дочерний модуль, у которого будет одно действие – «показать дорогу». А вот, например, модуль новостей будет иметь несколько дочерних модулей, так как каждый дочерний модуль – это своя новостная лента со своим административным интерфейсом (вам не придется разрабатывать административный интерфейс для каждого дочернего модуля) и своими действиями. Причем параметры этих действий могут быть разными.

Внешняя структура модуля

Прежде всего, давайте познакомимся со структурой любого модуля, используемого в CMS. Сразу хочу отметить, что здесь представлена структура модуля в разрезе взаимодействия администратора сайта и модуля. То есть, представлены те объекты (абстракции), через которые администратор сайта будет управлять модулем и настраивать его работу.

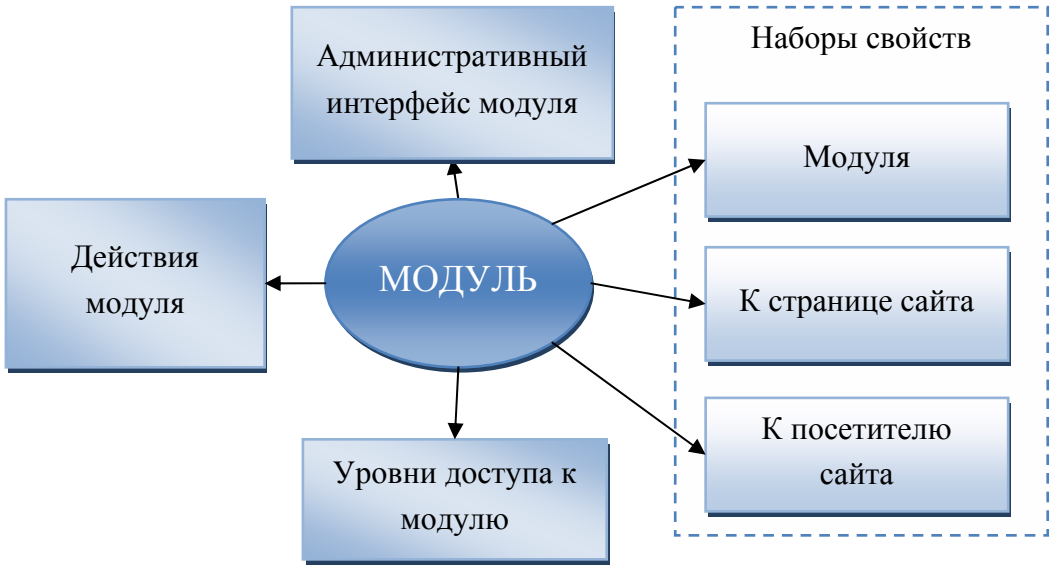

Рисунок 3: Схема объектов, которыми оперирует администратор сайта при разработке модуля.

Свойства модуля

Свойства модуля предназначены для управления общими настройками модуля, если в них есть необходимость. Значения свойства модуля определяются администратором сайта и могут быть заданы как у базового модуля (в этом случае значения свойств будут наследоваться дочерними модулями), так и у конкретного дочернего модуля.

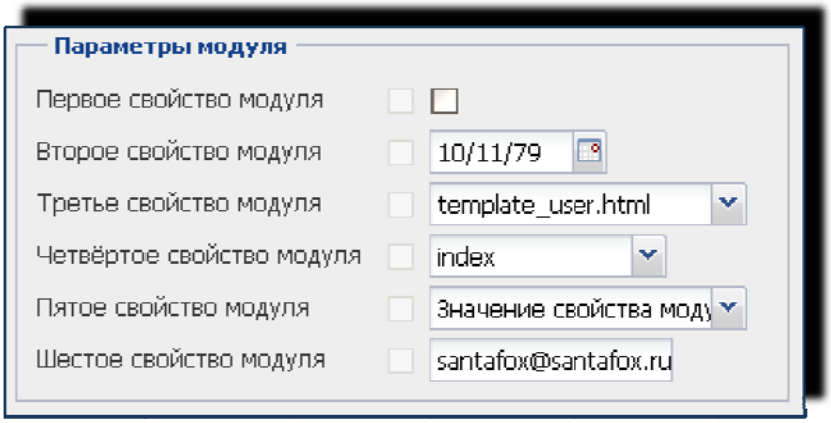

Рисунок 4. Пример внешнего вида редактора свойств модуля, то каким его видит администратор сайта

Модуль может иметь произвольное количество свойств, их количество и тип определяет разработчик модуля на стадии проектирования и корректирует на стадии разработки.

Типы свойств модуля стандартизированы CMS, помимо этого CMS отвечает за вывод этих свойств, для редактирования в административном интерфейсе, а также за получения значения свойства конкретным дочерним модулем.

Свойства модуля к странице сайта

Свойства модуля к странице сайта могут быть прописаны только дочерним модулем. Особенность данного вида свойств заключена в том, что значение этого свойства может быть задано у каждой страницы сайта.

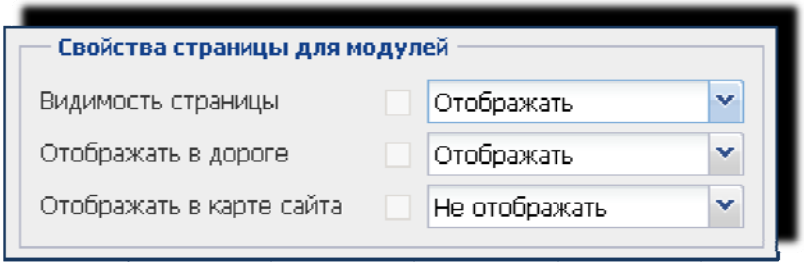

Рисунок 5. Пример внешнего вида редактора свойств модуля у страницы сайта

При получении значений свойств используется принцип наследования от родительской страницы. То есть, при обращении к значению свойства для конкретной страницы

Тип задаваемых свойств совпадают с типами свойств у модуля. Так же как и в свойствах модуля, разработчику необходимо лишь определить свойства модуля к странице, а за управление этими свойствами отвечает CMS.

Свойства модуля к посетителю сайта

Авторизация посетителей на сайте, а также хранение информации о посетителях осуществляется CMS. Однако, формирование различных форм, затрагивающих процесс авторизации и управления авторизированным пользователем, осуществляется отдельным модулем.

Каждый пользователь сайта, который может быть авторизирован на сайте имеет ряд заранее определенных полей. Помимо них, модулю, возможно, необходимо добавить свои поля, с тем, что бы запросить у пользователя сайта дополнительные данные (например: пол, любимый цвет, уровень доходов и т.п.)

При этом дополнительные параметры на пользователя может проставить как дочерний модуль, так и базовый (или одновременно). При этом соответственно, параметр от базового модуля добавиться только один раз, а параметров от дочерних модулей будет столько же, сколько проинсталлированных дочерних модулей. Для пояснения различий приведем следующий пример: для осуществления E-mail рассылок модулю новостной ленты необходимо знать электронный адрес для рассылки, а так же список новостных лент, на которые подписан пользователь. Для решения этой задачи, базовому модулю новостной ленты необходимо дописать к пользователю поле для хранения адреса рассылки, а каждому дочернему - поле для хранения признака подписки на данную новостную ленту.

Сейчас модуль имеет возможность добавлять к пользователю поля только строкового типа. В дальнейшем, возможные типы будут приведены к общему стандарту типов всех видов свойств, используемых модулем.

Административный интерфейс модуля

Большинству модулей необходимо иметь свой собственный административный интерфейс, где будет осуществляться непосредственное управление модулем. SantaFox™ предоставляет модулю самому конструировать административный интерфейс и управлять им, однако, SantaFox™ необходимо знать какой тип административного интерфейса будет использовать модуль. Возможны следующие варианты:

- у модуля нет административного интерфейса;
- модуль имеет один административный интерфейс, в не зависимости от количества дочерних модулей, проинсталлированных в системе;
- для каждого дочернего модуля, проинсталлированного в системе, вызывается свой экземпляр административного интерфейса.

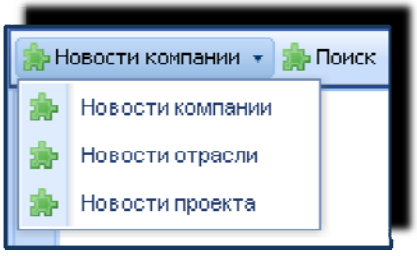

Рисунок 6. Модуль "Новости" имеет три дочерних модуля, у каждого из которых свой интерфейс администратора. Количество дочерних модулей у модуля "Поиск" не имеет значения, так как их интерфейс един.

При разработке административного интерфейса третьего типа следует быть внимательным. Необходимо помнить, что один и тот же код модуля, отвечающий за формирование административного интерфейса, вызывается для разных дочерних модулей. Для того, что бы различать какой конкретно дочерний модуль вызвал административный интерфейс необходимо использовать функцию ядра, возвращающую уникальный ID модуля.

Уровни доступа к модулю

SantaFox™ допускает множественное администрирование сайта. Для этих целей в CMS введены группы администраторов, которым можно открывать (закрывать) доступ к тем или иным функциональным возможностям CMS. Модуль так же может участвовать в процессе квотирования прав доступа администратору сайта.

Данная возможность необходима, например, сложному новостному модулю, когда перед тем как опубликовать новость, нужно получить «визу» корректора. Соответственно корректор – это администратор сайта, у которого есть права только на модуль новостной лента, и только на простановку там отметки «опубликовать» или «доработать»

Для того что бы различать разные уровни доступа модулю необходимо их ввести (обозначить) а затем пользуюсь функциями CMS проверять в необходимых местах кода наличие у конкретного пользователя значение этого права.

Следует отметить, что уровни доступа к модулям взаимосвязаны с типом его административного интерфейса, так как если административный интерфейс модуля не различает количество дочерних модулей (тип 1) то и уровни доступа прописываются только к базовому модулю. Если же административный интерфейс модуля вызывается для каждого дочернего модуля (тип 2), то и уровни доступа можно задать для каждого дочернего модуля.

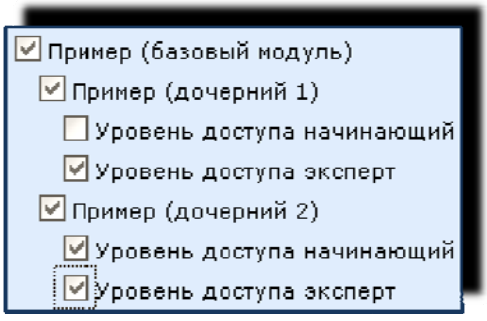

Рисунок 7. Пример модуля, у которого тип административного интерфейса "2" и два уровня доступа.

Действия модуля

Действия модуля – это собственно, то ради чего создаётся модуль. Действия модулей вызываются при формировании страницы сайта, и возвращенный ими HTML код замещает метки в шаблоне страницы.

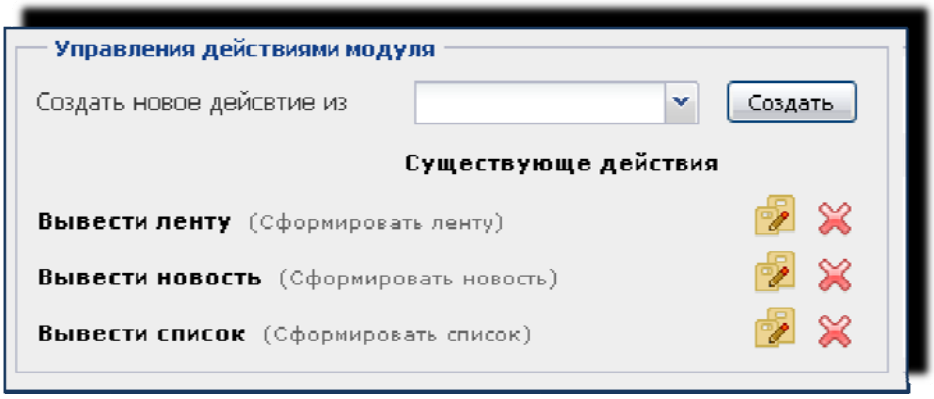

Рисунок 8. Три действия модуля "Новости" созданные из трёх разных методов модуля

С точки зрения модуля, действие – это обычный метод модуля, у которого администратор сайта может определить значения параметров. Именно действия (т.е. методы модуля с параметрами) будут вызываться ядром при обработке меток в шаблоне.

В качестве значений, передаваемых в такой метод, могут использоваться только стандартные типы значений используемых в CMS.

С чего начать разработку модулей

Прежде всего, следует определить функциональные возможности, которыми будет обладать модуль. После этого необходимо определить количество методов модуля, из которых могу конструироваться действия.

Что бы прояснить ситуаций с методами и действиями рассмотрим следующий пример. Есть модуль «Новости», у которого есть метод *news_lenta_show ($news_number)*. Данный метод отвечает за формирование новостной ленты из базы новостей. В качестве параметра ему необходимо передать число, показывающее, какое максимальное количество новостей должно быть отображено. При сборке сайта администратор создаст действие «*показать ленту*» и в качестве значения единственного параметра задаст, например цифру «4».

Затем необходимо продумать какие ещё параметры модуля пользователь должен контролировать. SantaFox™ дает модулю богатые функциональные возможности, потому, сначала необходимо определится с конкретным списком параметров, а затем уже распределить где и как эти параметры будут определяться.

Допустимые типы параметров

SantaFox™ поддерживает несколько типов параметров, используемых в свойствах модуля, свойствах модуля к странице, в параметрах действия. Весь процесс обработки этих параметров

берет на себя CMS. Разработчику модуля необходимо лишь обозначить их, и затем, получать значения этих параметров. Для создания параметра, необходимо использовать соответствующий этому параметру класс:

- properties_string (строка)
- properties_date (дата)
- properties_checkbox (галочка)
- properties_select (выпадающий список значений)
- properties_file (файл)
- properties_pagesite (страница сайта)

Создание любого параметра модуля выглядит следующим образом:

```
01.      $param = new <объект свойства>;
02.      $param->set_id('<идентификатор>');
03.      $param->set_caption('[#<языковая переменная>#]');
```

Помимо этого, в необязательном порядке может быть указано значение по умолчанию, для данного параметра:

```
01.      $param->set_default ('<значение по умолчанию>');
```

В зависимости от конкретного типа класса, параметры могут иметь дополнительные инструкции, которые перечислены ниже.

properties_file (файл)

Использует следующие дополнительные методы, для настройки параметра:

- **set_patch($path)** – задаёт путь к каталогу на сервере, из которого будет составляться список файлов, доступных для администратора сайта в качестве значения параметра.
- **set_mask($mask)** – задаёт расширения файлов, которые будут попадать в список для выбора. Если необходимо указать несколько расширений, они указываются через запятую.

properties_select (выпадающий список значений)

Использует следующие дополнительные методы, для настройки параметра:

- **set_data($data)** – устанавливает массив возможных значений для выбора. В качестве ключа массива используется возможное значение параметра, а в качестве значение массива – языковая переменная отображаемая пользователю

Приступаем к разработке

И так, надеемся, что у Вас сложилось понимание того, что собой представляет модуль в понятиях Santafox™. Теперь собственно можно начать писать код модуля. В этом разделе мы опишем этапы, которые нужно пройти, что бы создать свой первый модуль. Будем предполагать, что мы уже определись с функциональной частью модуля и его свойствами, которые будет редактировать администратор сайта.

Структура файлов

Все модули сайта собраны в одном каталоге *modules*. Для каждого из них отведена своя папка, названная в соответствие с уникальным идентификатором модуля. В каталоге модуля должны обязательно присутствовать следующее файлы:

- *install.php*
- *<id модуля>.class.php*, где в качестве *<id модуля>* указывается тоже идентификатор, который указан в качестве имени папки этого модуля;
- Директория *lang* с файлами языковых переменных. Обязательно наличие файла ru.php и en.php содержащие соответственно значения языковых переменных для русского и английского языка.

Идентификатора модуля – это идентификатор базового модуля, который придумывается разработчиком модуля и ни как не меняется средствами CMS.

Все файлы, используемые модулем и не относящиеся к компонентам CMS, должны находиться в его каталоге, за исключением тех файлов, которые могут относиться к контенту сайта. Для их хранения используется отдельная папка, вне модуля, в которую и должны помещаться такие фалы в случае необходимости.

Избегаете хаотичного размещения файлов в каталоге модуля. SantaFox™ не накладывает жестких ограничений на структуру файлов в каталоге модуля, однако мы рекомендуем придерживаться установленного порядка.

Структура файла install.php

Описание класса

Данный файл описывает основные данные модуля, которые необходимы CMS для корректной работы с модулем. Прежде всего, в этом файле необходимо определить класс, отвечающий непосредственно за инсталляцию и деинсталляцию модулей и его четыре функции.

```
01.      class <идентификатор>_install extends install_modules
02.      {
03.             function install($id_module)
04.             function uninstall($id_module)
05.             function install_children($id_module)
06.             function uninstall_children($id_module)
07.      }
```

Указанные функции класса определяют, какие действия должны производиться при инсталляции и деинсталляции базового и дочерних модулей.

В качестве параметра в каждую функцию передаётся идентификатор модуля, с которым осуществляется действие.

В функциях инсталляции необходимо поместить код, создающий таблицы базы данных, формирующий каких либо первоначальные данные и т.п. При этом следует помнить, что функция *install()* будет вызвана лишь один раз, в момент подключения базового модуля к системе, а функция *install_children()* будет вызываться каждый раз, при инсталляции нового дочернего модуля.

Не забывайте использовать *$id_module*, для того, что бы разделить данные дочерних модулей друг от друга, а также выделить данные базового модуля, среди остальных (если это необходимо).

Создание экземпляра класса

После того, как в экземпляре класса описаны процедуры инсталляции и деинсталляции, необходимо собственно создать объект *$install* и установить его параметры.

```
01.      $install = new <идентификатор>_install();
```

Все параметры задаются с помощью методов доступных у этого объекта. Порядок задачи параметров не важен, но мы рекомендуем сохранять структуру уже имеющихся файлов инсталляции других модулей, с тем, что бы поддерживать их единообразие.

Основные данные модуля

Основными параметрами модуля являются:

- идентификатор;
- название;
- тип административного интерфейса;

Задаются эти параметры следующим образом:

```
01.      $install->set_name('<языковая метка>');
02.      $install->set_id_modul(<идентификатор>);
03.      $install->set_admin_interface(<тип интерфейса>);
```

Функция *set_name('<языковая метка>');* задаёт имя базового модуля, которое будет отображаться администратору сайта. Указывается языковая переменная.

Функция *$install->set_id_modul(<идентификатор>)* задаёт идентификатор базового модуля. Он должен совпадать с именем папки модуля, и являться уникальным среди всех остальных модулей, используемых в CMS.

Функция *$install->set_admin_interface(<тип интерфейса>)* определяет тип административного интерфейса, который будет использовать данный модуль

- **0** – модуль не будет иметь собственного административного интерфейса.
- **1** – административный интерфейс модуля будет представлен на базовый модуль и не будет завесить от конкретного дочернего модуля.
- **2** – административный интерфейс будет различать разные дочерние модули и вести их раздельное администрирование.

Параметры модуля

```
01.      $param = new <объект свойства>;
02.      $param->set_id('<идентификатор>');
03.      $param->set_caption('<языковая метка>');
04.      $install->add_modul_properties($param);
```

В качестве < *объект свойства* > указывается имя объекта отвечающего за конкретное свойство того типа, который вам нужен.
set_id ('<идентификатор>') задаёт уникальный идентификатор свойства. По этому идентификатору, модуль будет обращаться к значению данного свойства. SantaFox™ сама отслеживает из какого дочернего модуля происходит обращение к свойству, и возвращает соответствующее значение. Значение идентификатора должно быть уникальным в пределах других свойств данного модуля.

В зависимости от типа свойства, возможно, необходимо задать ещё какие-то свойства параметров, подробнее о типах параметров можно узнать в соответствующем разделе.

Параметры модуля к странице

Каждый дочерний модуль может добавить какое-то свойство к страницам сайта. Свойство может быть одного из допустимых типов. Получения значения свойства происходит через функцию ядра. Добавляется свойство следующим образом:

```
01.      $param = new <объект свойства>;
02.      $param -> set_id('<идентификатор>');
03.      $param -> set_caption('<языковая метка>');
04.      $install -> add_page_properties ($param);
```

Как видно, описание непосредственно параметра абсолютно идентично с описанием параметра модуля, различия лишь в последних функциях.

Параметры модуля к посетителю сайта

У модуля есть возможность задавать дополнительные параметры к посетителям сайта (если на сайте установлен модуль авторизации). Описание самого параметра осуществляется стандартным способом. **В данной версии CMS поддерживаются параметры только одного типа String.**

```
01.      $param = new properties_string();
02.      $param -> set_id("sex");
03.      $param -> set_caption("Ваш пол");
```

Для регистрации этого параметра используется метод *add_user_properties()***.** Имеет следующий синтаксис:

```
04.      function add user properties ($param, $multi = false,
   $admin = false)
```

- **$param** – объект типа propertie_string.
- **$multi** – если TRUE - то этот параметр будет прописываться каждым экземпляром дочернего модуля, в противном случае только базовым модулем.
- **$admin** – если TRUE - то значит, доступ к этому параметру пользователя должен иметь только администратор, в противном случае и сам пользователь имеет доступ к этому свойству.

Таким образом, для примера с модулем новостей в котором есть подписка на новости, строчки кода будут выглядеть следующим образом:

```
01.      $param = new properties_string();
02.      $param->set_id("email");
03.      $param->set_caption("Адрес для рассылки");
04.      $install->add_user_properties($param);
05.
06.      $param = new properties_string();
07.      $param->set_id("subscribe");
08.      $param->set_caption("Признак подписки");
09.      $install->add_user_properties($param, true);
```

За формирования формы редактирования этих параметров отвечает отдельный модуль авторизации. Ядро обеспечивает выдачу, хранения и запись всех этих данных пользователя.

Публичные методы модуля

В теле модуля они ничем не отличаются от всех остальных методов, и любой метод модуля может стать публичным. Для этого необходимо описать его следующим образом:

```
01.      $install->add_public_metod('pub_show_menu', '<языковая
   метка>');
02.      $install-
   >add_public_metod_parametrs('pub_show_menu',$param);
03.      $install-
   >add_public_metod_parametrs('pub_show_menu',$param);
```

Сначала задается имя публичного метода (так, как оно задано в теле модуля), а потом указывается языковая переменная с названием этого метода, расшифровывающая администратору сайта назначение данного действия.
После этого указываются параметры, используемые методом. Параметры должны указываться в порядке их объявления в заголовке модуля.

Разграничение доступа

Если необходимо ввести разграничение доступа администраторов сайта к той или иной информации необходимо использовать следующие функции:

```
01.      $install->add admin acces label('<id>', '<языковая
   метка>');
```

< *id*> - уникальное ID права доступа, в пределах модуля, используемое для обращения к значению этого права.
<*языковая метка*> - названия этого уровня доступа для администратора сайта.

В дальнейшем, при необходимости проверки права, модуль использует функцию ядра, в качестве параметра которой передает ID права.

Заключение

Надеемся, что данное руководство дало вам всю необходимую информацию для разработки собственных модулей, работающих под управлением SantaFox™.