Система управления сайтами NetCat

версия 4.2

Руководство разработчика

Компания «НетКэт»
Москва, 2010 г.

Компания «НетКэт», отдел разработки
Телефон: (495) 783-60-21 (многоканальный)
Сайт: http://www.netcat.ru
Электронная почта: support@netcat.ru

Разработка системы управления сайтами NetCat: К. Хряпин, В. Островский, В. Бахреньков, Р. Сакал, Н. Громин, Д. Спирин, А.Наливайко, Д.Варин
Документация: Н. Громин, Д. Спирин, Д. Васильев, В. Бахреньков , Д.Варин
Руководство разработчика для версии 4.2.

Оглавление

Введение

Данное Руководство позволит вам создавать сайты с помощью системы NetCat, управлять ими после создания, изменять и расширять функциональность. Перед прочтением настоящего Руководства изучите Руководство пользователя, т.к. вам понадобятся базовые знания о системе, приведенные в нем.

Человек, создающий сайт на NetCat, должен:

-знать HTML в объеме, необходимом для создания обычного статического сайта;
-уметь работать с растровой компьютерной графикой, например, в пакете Adobe Photoshop (за исключением случаев, когда на сайте не используются графические элементы);
-представлять себе принципы работы реляционной СУБД (примеры реляционных СУБД: Microsoft Access, Microsoft SQL Server, MySQL).

Также желательно знать основы языка программирования PHP и языка SQL. Язык PHP похож на языки C, C++, Perl, поэтому если вы знаете какой-либо из этих языков, то скорее всего вам не потребуется серьезное изучение PHP.

Если вы хотите создавать собственные модули, вам необходимо:

-хорошо знать язык PHP, иметь опыт разработок на нем;
-хорошо знать SQL, его реализацию в СУБД MySQL.

Также подразумевается, что разработчик сайта должен быть «продвинутым пользователем» компьютера («advanced user»). Если сайт на NetCat создает не один человек, а команда разработчиков (например, веб-студия), эти знания и навыки могут быть распределены между ее участниками.

Часть 1. Описание системы

Технические требования

Минимальные аппаратные требования для системы NetCat:

- ✓Компьютер с процессором Pentium 166 МГц (рекомендуется от 300 МГц)
- ✓Оперативная память 64 Мб (рекомендуется от 256 Мб)
- ✓Место на жестком диске 15 Мб

Программные требования для системы NetCat:

- ✓Операционная система Microsoft Windows или Unix (Linux, FreeBSD и пр.);
- ✓Веб-сервер Apache 1.3.30 и выше;
- ✓PHP 5.1 и выше (может быть собран как модуль Apache или как CGI-скрипт);
- ✓СУБД MySQL 4.1 и выше.

Идеологические требования к сайтам

Использование NetCat накладывает некоторые ограничения на сайты, которые будут работать под ее управлением. В большинстве своем ограничения несущественные, однако, их стоит иметь в виду, создавая сайты на NetCat. Ниже приведен список ограничений.

Иерархическая древовидная структура сайта

Ограничение весьма условно – в настоящий момент редко встречаются сайты с другим видом структуры. Это ограничение означает, что структура сайта под управлением NetCat должна иметь вид дерева, например, как на представленном ниже фрагменте структуры:

- *Каталог*
  - *Валенки*
  - *Дубленки*
  - *Лобзики*
- *О компании*
  - *Миссия*

- *Менеджмент*
  - *Руководство компании*
  - *Руководители отделов*
- *Новости*
- *Вакансии*

Партнерам

- *Регистрация*
- *Принципы работы*

Также возможна неполная имитация сетевого вида структуры, когда один дочерний элемент структуры может иметь несколько родительских.

Разделение каждой страницы сайта на 3 части

Каждая страница сайта делится на три условных части: header (хедер, верхняя часть страницы), содержательная часть страницы и footer (футер, нижняя часть страницы). При этом термины «верхняя» и «нижняя» не означают дословное геометрическое понимание – подразумевается HTML-текст до и после содержательной части страницы. На футере и хедере могут располагаться элементы оформления сайта, навигации, вспомогательные элементы (баннеры, опросы и пр.). Ниже приведен схематический пример простой страницы: белым цветом обозначена содержательная ее часть, светло-серым – хедер и темно-серым – футер.

<table>
<tr><td>ЛОГОТИП</td><td colspan="2">Компания «Стратосфера»
Бесспорно, Большая Медведица параллельна.

О компании Продукция Дилеры Поставщики Форум</td></tr>
<tr><td rowspan="2">О компании:
-Структура компании
-Руководство компании
-Новости
-Филиалы</td><td colspan="2">Новости</td></tr>
<tr><td>30.11.07. Цицерон говорит также в трактате "О старости" (De senectute).
23.09.07. У планет-гигантов нет твёрдой поверхности.
15.07.07. Атомное время отражает непреложный Ганимед.</td><td>См. также:
-example.ru
-example.net
-example.com</td></tr>
<tr><td colspan="3">© ООО «Стратосфера» info@example.ru</td></tr>
</table>

Впрочем, на странице может быть гораздо больше информационных блоков. В данном Руководстве описаны способы создания довольно сложных структур страниц при помощи NetCat.

Использование Macromedia Flash

NetCat позволяет использовать flash-заставки, как и любые другие файлы, используемые в HTML-коде. Однако в случае создания полноценного flash-сайта NetCat может быть использован с весьма существенными ограничениями. Так, если система навигации по сайту под NetCat реализована на flash, для интеграции flash-роликов с системой управление структурой NetCat потребуются дополнительные доработки ролика. То же касается случаев, когда сайт представляет собой единый flash-ролик.

Использование фреймов (frames)

В настоящее время фреймы встречаются очень редко – это связано с многочисленными проблемами, которые возникают при их использовании. NetCat содержит встроенные средства создания сайтов с традиционной, «бесфреймовой» навигацией. При помощи NetCat также можно реализовать фреймовый сайт, но это потребует использования некоторых специальных приемов.

Подход к данным как к спискам шаблонной информации

Все данные, выводимые в содержательной части страниц, в NetCat представляются в виде списков некоторых шаблонных элементов. Это могут быть списки товарных позиций, новости, списки сообщений в форуме, более сложные конструкции. Описываются данные компонентами. Простейшим компонентом является HTML-блок текста. С его помощью реализуются все задачи, для которых неприменим «шаблонный» подход. Так, неструктурированный HTML-текст (с картинками, flash-роликами, скриптами JavaScript и пр.) обычно представляется как список данных, состоящий из одного элемента компонента «HTML-текст». Эти элементы называются объектами. Списки объектов на странице могут образовывать составные конструкции, например, новости с комментариями, личные сообщения между пользователями и пр.

Часть 2. Архитектура системы

Систему NetCat можно разделить на 2 взаимосвязанные части: Систему администрирования (back-office) и Систему ввода-вывода (front-office). Первая отвечает за управление сайтом и отдельными сущностями в его рамках: структурой, правами, компонентами и пр. Фактически Система администрирования является специализированным интерфейсом к базе данных системы. Вторая часть отвечает за формирование HTML-страниц в браузере пользователя и за вывод информации. Система ввода-вывода является интерфейсом посетителя, т.е. посетитель сайта видит результаты работы непосредственно этой системы. Обе системы работают с единой базой данных.

При разработке системы NetCat сущность «сайт» разделяется на несколько структурных составляющих, которые необходимо знать при создании сайта. Итак, любой сайт в представлении NetCat является совокупностью следующих сущностей:

1. Основные сущности, на которые можно разделить любой стандартный сайт
    1.1. Структура
    1.2. Компоненты (функциональные элементы, отвечающие за отображение содержимого страниц)
    1.3. Макеты дизайна страниц сайта
    1.4. Данные (текстовые, графические и пр.)
2. Дополнительные сущности, связанные с разграничением прав
    2.1. Пользователи и права доступа
3. Дополнительные сущности, введенные при проектировании системы
    3.1. Вспомогательные данные (списки)
    3.2. Дополнительные программные модули

Структура сайта

Основной структурной единицей в системе является сайт. Каждому сайту соответствует собственный домен, например, example.ru, inside.example.ru. Поэтому в стандартных задачах сайт только один. Дополнительные сайты следует использовать в том случае, если нужно выделить в отдельный «подсайт» некий объем информации, или на одной копии системы поддерживать несколько сайтов.

При поддержке нескольких сайтов на одной копии системы следует помнить, что лицензионное соглашение NetCat разрешает поддерживать несколько

сайтов на одной копии только в том случае, если все она фактически принадлежат одному юридическому или физическому лицу – владельцу копии NetCat.

Внутри сайта структура определяется системой разделов. Каждый раздел может иметь неограниченное количество подразделов, которые, в свою очередь, могут делиться на подразделы. Так реализуется иерархическая структура сайта. Совокупность включенных разделов образует карту сайта.

Каждый раздел имеет несколько свойств (атрибутов), которые влияют на права доступа к страницам данного раздела, внешний вид страниц, свойства навигации и пр. Состав свойств может дополняться при помощи интерфейса редактирования системной таблицы «Разделы» (см. раздел «Системные таблицы» меню «Разработка»). Большинство свойств характерно как для сайта, так и для раздела.

Для того чтобы иметь возможность наполнять раздел информацией, необходимо указать компонент (или несколько компонентов), который будет обеспечивать пользователю возможность добавления данных, изменения, а также отображения этих данных. Простейший компонент – обычный HTML-текст, более сложные – каталоги товаров, простой форум, несложные интерактивные элементы, таблицы данных с возможностью поиска по ним, веб-формы и пр.

Также в системе управления структурой реализовано важное свойство – наследуемость атрибутов. Это означает, что, например, если для некоторого раздела указаны определенные настройки (макет дизайна, уровни доступа, логотип и пр.), то для его подразделов эти свойства будут такими же, если их специально не переопределить. Это позволяет максимально сократить время на ввод и настройку структуры сайта.

Компоненты

Компонент – это функциональная сущность, которая управляет содержательной частью страницы. Целесообразность введения компонентов можно понять на следующем примере. Допустим, на сайте есть каталог товаров. Для каждого товара определено название, фото, описание и цена (на практике свойств зачастую гораздо больше). При добавлении или редактировании позиции товара (без использования компонентов) необходимо выложить изображение товара на сервер и ввести/отредактировать HTML-код товара. При использовании компонента «Каталог товаров» достаточно ввести значения в HTML-форму и закачать картинку через веб-интерфейс.

Кроме того, использование компонентов позволяет быстро и просто осуществлять следующие действия (на примере того же каталога товаров):

- поиск/выборку по базе товаров (NetCat содержит встроенные средства поиска по любым атрибутам – цена, название, категория и пр.);
- изменения в формате вывода позиций (достаточно изменить HTML-код в настройках компонента – и он изменится во всем каталоге);
- реализацию разных форматов вывода данных за счет альтернативной разметки (в зависимости от каких-либо параметров);
- проверку на правильность ввода (например, не допускать ввод букв в поле «Цена»);
- подключать новые функциональные возможности (например, заказ товара или покупку через платежную систему);
- реализовывать импорт/экспорт объектов (особенно это важно при больших объемах информации – имея на входе файл Microsoft Excel, можно получить готовый каталог товаров на сайте) и т.д.

Система управления компонентами сочетает мощь и гибкость: с ее помощью можно реализовать огромный объем задач, в то же время она проста и удобна в использовании.

Для каждого компонента данных определяются:

1. набор полей (например, для простейшего каталога товаров это может быть «Название», «Описание», «Картинка», «Цена»);
2. шаблоны вывода содержимого страниц (4 шаблона: префикс, суффикс, макет вывода (для списка объектов) и макет полного вывода (если каждому объекту будет соответствовать своя отдельная страница);
3. шаблоны различных действий (добавления/изменения/удаления, действия после добавления/изменения, подписки и пр.);
4. действие по умолчанию (что пользователь видит при заходе на страницу: список объектов, веб-форму для добавления данных или их отправки, форму поиска по данным);
5. пользовательские настройки (возможность конечному пользователю настраивать вид отображаемых данных без необходимости владения специальными навыками);
6. шаблоны компонентов (использование абсолютно других префикса, суффикса и всех остальных полей компонента для вывода одних и тех же данных).

В составе NetCat поставляется большое количество готовых компонентов, но разработчик может создать и собственные компоненты. Несколько типов функциональных элементов, которые можно реализовать при помощи системы:

- структурированный список: новости, статьи, фотогалереи, простые форумы, каталоги товаров и пр. Разработчик определяет состав полей и шаблон отображения, а формы добавления/изменения с проверкой валидности, удаление, листинг по страницам строятся автоматически. Конечный пользователь может настраивать параметры отображения;
- простые сервисные функционалы типа несложной системы показа баннеров, файл-менеджера, личных сообщений между пользователями;
- несложные интерактивные элементы: вопросы-ответы, гостевые книги, простые форумы и блоги, комментарии к материалам;
- комбинированные компоненты, например, новости с комментариями, отели с номерами и пр.;
- формы поиска по данным: по подстроке, диапазону значений для численных атрибутов, точное соответствие, выбор из списка;
- веб-формы (отправка на почту заполненной формы на почту, хранение и отображение введенных данных, статистика и пр).

Компоненты также применимы для тех случаев, когда списочный вывод не требуется. В этом случае используется один объект компонента «HTML-текст».

На одной странице могут быть использованы несколько компонентов, например, новость и комментарии посетителей к ней; список вакансий компании, выше которого указан вводный текст, а ниже – список менеджеров и т.д.

Макеты дизайна

Как уже показано выше, каждая страница делится на 3 части: хедер, футер и содержательную часть. Формат вывода содержательной части определяется компонентом, а ее расположение на странице, равно как и футер с хедером – макетом дизайна.

Основная практическая функция макета дизайна (не считая самого оформления страницы) – реализация навигации по сайту. Макет обеспечивает вывод навигации различных уровней, внешний вид которой может настраиваться, и различных видов: оглавление сайта, меню 2-3-4-… уровня, «хлебные крошки» (Рога и Копыта > О компании > Руководство компании) и пр. Другой немаловажной особенностью NetCat является тот факт, что внешний вид элементов также может настраиваться: это могут быть текстовые ссылки на

разделы, выпадающие списки (тег <select>), изображения, flash-ролики, выпадающие DHTML-меню, а также сочетания этих видов (иконка + название раздела и пр.).

Сам макет представляет собой HTML-страницу, конвертированную во внутренний формат NetCat. Процесс конвертации несложен. Так, если исходный макет подразумевает оглавление сайта в определенном месте страницы, оформленное каким-то образом (таблицы, выпадающие меню, картинки и пр.), то в конечном макете этот HTML-код заменяется на функцию s_browse_level(), а внешний вид навигации настраивается в специальном поле «Шаблоны вывода навигации».

Макетов в системе может быть несколько. В большинстве случаев их два: для титульной страницы и для внутренних страниц (возможна также версия для печати). Однако можно создать неограниченное количество макетов и использовать разные макеты для разных разделов (по умолчанию все страницы сайта имеют тот макет, который определен для всего сайта, но на любом уровне структуры его можно переопределить). Структура свойств макета также редактируется через интерфейс управления системными таблицами.

В системе макетов реализована наследуемость, что позволяет сократить время настройки дизайна и предоставить разработчику более удобный интерфейс работы. Так, если есть задача использовать различную цветовую палитру для разных разделов, необходимо:

- добавить в системную таблицу «Макеты дизайна» поле «Таблица CSS»;
- создать и заполнить макет по умолчанию;
- создать дочерний макет, в котором заполнить только поле «Таблица CSS»;
- определить в настройках нужных разделов созданный дочерний макет.

При работе с макетами стоит учесть, что кавычки "" являются спецсимволом PHP. Например, ими мы выделяем функцию:

```
<td>".s_browse_level(0, $browse_sub[0])."</td>
```

Поэтому использовать их в HTML коде нужно специфически: кавычки необходимо экранировать, т.е. перед каждой кавычкой поставить обратный слэш \". В разделе Инструменты в системе администрирования есть специальный функционал для автоматического экранирования кода. Для упрощения работы мы рекомендуем Вам использовать одинарные кавычки ‘

(они равноправны двойным практически во всех стандартах, включая HTML, XHTML) .

Пользователи и права доступа

С помощью системы управления правами можно решать огромный спектр задач: электронные СМИ и сообщества, интернет-магазины различных типов, закрытые торговые и дискуссионные площадки, системы работы с удаленными партнерами и сотрудниками и многое другое.

Система управления правами состоит из двух условных составляющих – прав, выданных конкретным пользователям или группам пользователей, и свойств сайта/раздела/компонента, определяющих их взаимодействие с различными категориями пользователей.

Права пользователя наследуются параллельно наследованию в структуре сайта. Т.е. пользователь, имеющий какие-либо права на сайт, автоматически получает аналогичные права на все разделы и подразделы внутри этого сайта; пользователь, имеющий какие-либо права на раздел, имеет аналогичные права на все компоненты раздела плюс права на все подразделы этого раздела.

Свойства доступа сайта/раздела/компонента

Каждый сайт/раздел/компонент раздела имеет 6 свойства, определяющих принцип доступа на операции:

1. просмотр;
2. добавление;
3. изменение собственных объектов;
4. включение/выключение собственных объектов;
5. удаление собственных объектов;
6. подписка (при наличии модуля «Управление подписками»).

Каждое из этих свойств является наследуемым (т.е. если на каком-либо уровне структуры оно не определено, ему присваивается значение этого свойства с верхнего уровня, которое, в свою очередь, определяется так же). Каждое свойство, определяющее права, может иметь значения:

1. доступ для всех – любой посетитель сайта может произвести эту операцию;

2. доступ для зарегистрированных пользователей – только зарегистрированные пользователи могут производить эту операцию;
3. доступ для уполномоченных пользователей – эту операцию могут производить только те пользователи, которым присвоены права на это.

Допустим, для раздела установлен следующий набор свойств на права:

- Просмотр – доступ для всех;
- добавление – зарегистрированные пользователи;
- изменение собственных объектов – уполномоченные пользователи;
- подписка – уполномоченные пользователи.

В этом случае:

- просматривать страницы раздела могут все посетители сайта, авторизация не будет производиться;
- добавлять объекты могут все зарегистрированные пользователи (кроме выключенных). Перед добавлением должна быть произведена авторизация – если пользователь ее не пройдет, объект не будет добавлен.
- изменять собственные объекты могут только зарегистрированные пользователи, у которых есть права на изменение собственных записей либо в этом компоненте раздела, либо в этом разделе, либо в любом разделе, иерархически находящемся «над» данным в структуре, либо в каталоге.
- подписка – то же самое.

С системой управления правами также связано свойство «Публикация объектов». Оно может иметь одно из значений – «после проверки администратором» (после добавления объекта он остается выключенным, а его публикация производится после включения) и «после добавления» (публикация добавленного объекта производится сразу по факту).

Типы прав пользователей

Каждый пользователь в системе может быть наделен неограниченным количеством прав. Кроме того, можно объединять пользователей в группы, которые, в свою очередь, также могут иметь определенный набор прав. Этот набор наследуется всем пользователям группы. Каждый экземпляр прав может быть одного из следующих типов:

1. директор (имеет полные права на систему);

2. супервизор (имеет полные права на систему, за исключением операций с директорами);
3. редактор;
4. администратор списка;
5. ограничение в правах;
6. гость.

Рассмотрим подробнее некоторые типы прав.

Редактор

Пользователь с правом «Редактор» имеет возможность управлять структурой сайта, его содержимым, а также при включении соответствующей опции, другими пользователями, имеющими права ниже его собственных.

При назначении какому-либо пользователю права «Редактор\Модератор» требуется указать участок, который он будет редактировать. Это может быть сайт (определенный или все сразу), раздел, либо компонент в разделе. Так же необходимо определить возможности в выбранной сущности:

- просмотр информации в каталоге/разделе/компонента раздела;
- добавление объектов;
- подписка на добавленные объекты (при наличии соответствующего модуля);
- изменение собственных объектов (т.е. записей, которые добавил данный пользователь);
- включение собственных объектов;
- удаление собственных объектов;
- модерирование (редактирование, включение/выключение, удаление всех объектов);
- администрирование (редактирование настроек; для сайта и раздела – управление компонентами в разделе/сайте).

Возможности независимы в том плане, что возможность, например, «Администрирование» не подразумевает «Модерирование», т.е. пользователь только с возможностью «администрирование» не может изменять объекты или подписаться на компонент в разделе, если в настройках компонента в разделе стоит «только «уполномоченным»

Пользователю с типом прав «Редактор\Модератор» можно разрешить управлять другими пользователями, а именно:

1. Добавлять пользователей.

2. Редактировать пользователей (в том числе и изменять некоторые права).
3. Удалять (модерировать) пользователей.

Названные действия могут быть применены только к пользователям с типом тип прав ниже, чем у Редактора\Модератора. Другими словами, Редактор\Модератор не сможет изменить права у директора, супервизора, разработчика или другого Редактора\Модератора.

Администратор списка

Пользователь с таким типом прав имеет доступ к спискам. Можно дать возможность пользователю управлять определёнными списками (включая системные), а так же назначить права на все списки сразу.

Возможности работы со списком:

1. Изменение — позволяет изменять элементы списка.
2. Добавление — позволяет добавлять элементы в список.
3. Модерирование — позволяет удалять элементы списка, переименовывать его, менять приоритеты элементов списка.

Если пользователю назначены права на модерирование всех списков, то он может так же добавлять новые списки, либо удалять существующие.

Во время назначения прав на редактирование списка, при выборе самого списка, некоторые из них отображены красным цветов. Это системные списки. По умолчанию никто не имеет к ним доступа (даже директор). Чтобы дать пользователю возможность редактировать системный список, нужно указать этот список явно.

Внимание! Не стоит присваивать пользователям права на редактирование системного списка без ЯВНОЙ необходимости!

Ограничение в правах

В системе NetCat существует возможность ограничить права определенному пользователю, т.е. запретить ему выполнять различные действия: (просмотр, добавление, изменение, подписка) как на сайте (включая все сайты в системе), так и в разделе, либо в компоненте раздела.

Ограничение на действие будет срабатывать, если в настройке действия доступа компонента в разделе указано «Зарегистрированные» или «Уполномоченные». При этом тип «Ограничение в правах» является приоритетным – даже если пользователь является администратором сайта, при присвоении ему типа «Ограничения в правах» он не сможет выполнять ограниченные действия.

Присвоение прав

Присвоение прав производится через систему администрирования директором, супервизором, либо Редактором\Модератором (имеющим соотв. полномочия). В окне присвоения прав сначала выбирается тип прав: «Директор», «Супервизор», «Редактор», «Администратор списка», «Ограничение в правах», либо гостевой доступ.

В случае выбора типа «Редактор\Модератор», «Разработчик» или «Ограничение в правах» требуется выбрать из выпадающего списка сущность, в отношении которой данный тип прав будет применен.

Для разрешения Редактору\Модератору управлять пользователями, необходимо поставить галочку (включить checkbox) напротив пункта «Управление пользователями» и, при необходимости, выбрать возможные действия (добавление, изменение, модерирование). Фактически, в этом случае у Редактора\Модератора появятся два права: одно на управление пользователями, второе – на управление указанной сущностью (сайтом, разделом, компонентом в разделе). Если вам требуется задать право только на управление пользователями, тогда нужно оставлять checkbox-ы (просмотр, добавление, изменение, подписка, модерирование, администрирование) пустыми.

Использование системы разграничения прав

При помощи сочетания системы разграничения прав и параметра публикации объектов можно организовать, например, систему публикации информационных материалов, когда несколько авторов могут добавлять в раздел материалы, которые проверяются корректором/редактором и только после этого публикуются. Для создания такого функционала обычно выполняются следующие операции:

1. Создается раздел с настройками доступа на чтение для всех, а на добавление, модерирование и изменение собственных объектов – для уполномоченных пользователей. К разделу подключается

соответствующий компонент (статьи, обзоры и пр.). Объекты после добавления проверяются администратором.

2. Создается группа пользователей «Авторы», которая наделяется правами добавления в созданный раздел. Каждый автор помещается в эту группу.

3. Создается пользователь с условным названием «корректор» или «редактор» и наделяется правами на модерирование данного раздела.

После этого все пользователи, входящие в группу «Авторы», смогут добавлять объекты в раздел (кроме того, пользователям из других групп можно прописывать права, аналогичные правам группы «Авторы»), но публиковаться эти материалы будут только после того, как «корректор» их включит. Если же наделить «корректора» правом на добавление, он сможет добавлять материалы самостоятельно.

В стандартных задачах весь спектр возможностей системы управления правами используется не часто. Обычно права (на просмотр – для всех, на другие операции – для уполномоченных пользователей) проставляются только для сайта, а во всех разделах эти параметры остаются неопределенными, то есть наследуются.

Основные действия в административной области и требуемые для этого права

Действие	Право не ниже
Действия со структурой	
Создание, удаление, изменение приоритетов сайтов	Администратор всех сайтов
Изменение настроек сайта	Администратор сайта
Добавление, удаление корневых разделов, изменение их приоритетов	Администратор сайта
Добавление, удаление подразделов раздела, изменение приоритетов подраздела	Администратор родительского раздела
Изменение настроек раздела	Администратор раздела
Добавление, удаление компонентов в раздел, изменение приоритетов компонентов	Администратор раздела
Изменение настроек компонента в разделе	Администратор компонента в раздела
Разработка	

Все действия с компонентам, макетами дизайна, системными таблицами	Супервизор
Добавление, удаление списков	Администратор всех списков с возможностью модерирования
Добавление элементов в список	Администратор списка с возможностью добавления
Изменение элементов списка	Администратор списка с возможностью изменения
Удаления элементов списка	Администратор списка с возможностью модерирования
Управление пользователями	
Просмотр пользователей	Модератор
Добавление пользователей	Модератор с возможностью добавления
Редактирование пользователей	Модератор с возможностью изменения
Удаление пользователей	Модератор с возможностью модерирования
Прочее	
Установка патчей, изменение настроек системы, работа с модулями и т.д.	Супервизор

Данные

Для управления данными необходимо в системе администрирования войти в меню выбранного раздела (к которому должен быть прикреплен минимум один компонент) и нажать на закладку «редактирование» (она выводится по умолчанию). На открывшейся странице появится список объектов, около каждого из которых будет показан блок меню работы с данным объектом – его атрибуты, ссылки «изменить», «удалить», «включить/выключить». Место в объекте, где показывается этот блок, определяется переменной $f_AdminButtons в макете вывода объекта данного компонента. Также на странице будут кнопки «добавить» и «удалить все».

Сами данные хранятся в таблицах MessageXX, где XX – номер компонента, который соответствует данной таблице (за исключением полей типа File – они закачиваются на сайт через веб-интерфейс). Структура таблиц редактируется при помощи интерфейса управления полями компонента. Так, если в компонент номер 4 добавляется поле, система модифицирует структуру таблицы Message4 при помощи SQL-оператора «ALTER TABLE». Аналогичная операция выполняется при изменении типа поля или при его удалении. Когда

создается новый компонент, в базе данных автоматически появляется новая таблица, к примеру, Message9, если номер компонента – 9.

Таким образом, если у компонента номер 4 есть поля «Name» (имя), «Address» (адрес), «Age» (возраст), «Sex» (пол), то в таблице Message4, помимо служебных, есть такие же поля.

Все объекты компонента номер XX хранятся в таблице MessageXX вне зависимости от того, в каком разделе находятся эти объекты. При выводе содержимого некоторого раздела происходят примерно следующие действия:

1. по компоненту раздела определяется номер используемого компонента;
2. по номеру компонента определяется таблица, в которой хранятся данные;
3. происходит выборка всех записей этой таблицы, атрибуты «Номер раздела» и «Номер компонента раздела» которых совпадают с текущим разделом и компонентом в нем;
4. если заданы дополнительные аргументы для выборки, производится дополнительная выборка в соответствии с параметрами запроса (например, выводить все объекты, где возраст от 20 до 30);
5. выводится префикс списка объектов для данного компонента, определенный макетом вывода префикса (см. настройки компонента);
6. если количество выбранных объектов больше нуля, то каждый объект, подходящий под указанные выше условия, форматируется (в соответствии с шаблоном вывода объекта в списке для данного компонента) и выводится;
7. выводится суффикс списка объектов для данного компонента.

Если на странице должна отобразиться только конкретная запись, она выводится в соответствии с шаблоном полного вывода данного компонента.

Списки

Списки – простейшая таблица, структура которой состоит из пяти полей: номер (ID) записи, имени (текстовое поле), приоритета, дополнительного значения и метки доступности (включен/ выключен). Так, если мы создаем список «Деньги» (Money), в нем будут пять полей: Money_ID, Money_Name, Money_Priority, Value, Checked. В теории баз данных термину «список» почти точно соответствует термин «классификатор».

Списки предназначены для использования в компонентах и в системных таблицах. Поясним использование списка в компоненте на примере. Пусть нам необходимо создать компонент «Сотрудники», каждая запись которого, помимо других полей, будет иметь поле «Пол». Для этого необходимо:

1. создать список «Пол» (английское название, к примеру, Sex) и добавить в него элементы «мужской» и «женский»;
2. создать компонент (например, номер 8) и его поля. При этом будет создана таблица Message8;
3. создать поле «Пол» (комментарий к полю) с названием, например, Sex. В «Типе поля» необходимо выбрать значение «Список», в «Формате» указав значение «Sex» – аналогичное названию классификатора. При этом в таблице Message8 появится поле Sex целого типа.

После этого при добавлении/изменении записей данного компонента для каждого объекта будет выводиться выпадающий список (HTML-тег <select>), в котором будут значения «мужской» и «женский».

Важной составляющей списка является возможность его сортировки. По умолчанию записи при выводе сортируются по их ID (по возрастанию). Однако в системе администрирования можно изменить этот порядок на сортировку по Имени или по Приоритету, а также выбор порядка сортировки: по возрастанию или по убыванию.

В компоненте при использовании поля типа Список можно получить как название элемента списка, так и его ID и дополнительное значение. Например, поле называется List, в таком случае $f_List вернет название, а $f_List_id – ID элемента, т.е. суффикс _id помогает получить ID, суффикс _value помогает получить дополнительное значение элемента.

При использовании альтернативных форм добавления/изменения объектов для вывода списка используйте функцию nc_list_select(). Подробнее она описана в конце этого руководства в Приложении 2.

Импорт списка

Для импорта списка необходимо в соответствующем разделе заполнить все поля формы. Список создается автоматически, Вам не нужно готовить его заранее.

Название таблицы (латинскими буквами) – название списка для использования в системе (например, Money, Type...).

Название - русскоязычное название списка.

CSV-Файл - текстовый файл с разделителем. Разделитель – ; (точка с запятой).

Сортировка - можно оставить как есть. Установится выбранный вариант сортировки.

Если в импортируемом файле только одна колонка, то она считается полем Элемент, если две – то первая колонка это Элемент, а вторая Приоритет. ID элементов проставятся автоматически при занесении в БД.

Модули

Модули поставляются в виде TGZ-архивов, содержащих программные файлы и наборы SQL-операторов. Модули используются для расширения функциональности и создаются в тех случаях, когда:

- необходима функциональность для ввода/вывода данных, которую нельзя реализовать при помощи стандартных механизмов управления компонентами. В этом случае модуль обычно содержит SQL-код для добавления компонентов и файл с необходимыми функциями;

- необходима реализация функционала, использующего компоненты, но требующего нестандартных операций с данными (например, корзина в интернет-магазинах). В этом случае, помимо SQL-операторов для создания компонента «Интернет-магазин», модуль содержит скрипт или набор скриптов, обеспечивающих работу с функционалом;

- необходима интеграция с каким-либо сервисом на сервере или вне его. Это может быть как поисковый движок, так и удаленный сервис (например, офисная торговая система). В этом случае модуль не сможет работать самостоятельно; необходимо будет также настроить внешний сервис. Инструкции по его настройке будут приложены к модулю.

Обычно после установки модуля его нужно настроить. Необходимые настройки доступны для редактирования в разделе «Модули» меню «Настройки системы». Значения настроек приводятся в документации к нему.

Установка модулей может происходить одним из двух способов: через веб-интерфейс и в терминальном режиме. Доступность этих способов зависит от настроек сервера, на котором размещается система. Подробнее об установке модулей см. «Руководство пользователя».

Часть 3. Перед тем, как начинать разработку

Установка системы

Установка системы производится при помощи инсталляционной программы, входящей в поставку NetCat. Однако, возможны случаи, когда программа не может автоматически установить систему, например:

- хостинг-площадка, куда производится установка, имеет нестандартные настройки;
- ваш компьютер работает под управлением операционной системы, отличной от Microsoft Windows.

В этом случае система должна быть установлена вручную. Эта процедура описана ниже.

Система NetCat может работать как под управлением операционной системы Microsoft Windows (Windows 98/200/NT/XP/Vista/7), так и под управлением Unix-систем (Linux, FreeBSD, AIX, Solaris и пр.). Стоит отметить, что чаще всего NetCat используется именно под операционными системами Unix, т.к. абсолютное большинство хостинг-провайдеров (компаний, которые осуществляют услуги по размещению сайтов в Интернете) используют на своих серверах именно эту систему. На компакт-диске приведен список крупнейших российских хостинг-провайдеров с дополнительными инструкциями по установке системы для каждого из них.

Однако для работы с системой вам не понадобится знать систему Unix, т.к. большинство хостинг-провайдеров позволяют осуществить в режиме визуального редактирования все операции, необходимые для установки системы. Установить систему и работать с ней в дальнейшем вы можете с удаленного компьютера, работающего под любой операционной системой. Единственное требование к рабочему компьютеру – наличие выхода в сеть Интернет.

Вы можете также использовать NetCat на домашнем или рабочем компьютере (в т.ч. под управлением системы Microsoft Windows). Это может быть актуально, в частности, в случаях, когда необходимо разработать сайт на локальном компьютере, заархивировать его и опубликовать в Интернете уже в готовом виде.

Системные требования

Минимальные аппаратные требования для системы NetCat:

1. компьютер с процессором Pentium 166 МГц (рекомендуется от 300 МГц);
2. оперативная память 64 Мб (рекомендуется от 256 Мб);
3. место на жестком диске 15 Мб.

Как вы могли заметить, система NetCat нетребовательна к системным ресурсам, поэтому, как правило, требования к компьютеру у нее такие же, как и у операционной системы.

Для работы системы NetCat на компьютере/сервере должны быть установлены следующие программные средства:

- веб-сервер Apache 1.3.30 и выше;
- PHP 5.1 и выше (может быть собран как модуль Apache или как CGI);
- СУБД MySQL 4.1 и выше.

Обычно все эти средства входят в поставку операционных систем семейства Unix и присутствуют на хостинг-серверах под этой ОС. Для Microsoft Windows их необходимо скачать и установить. Все они являются бесплатными и находятся в свободном доступе в сети Интернет. Существуют программные пакеты, куда включены эти средства, например, пакет «Денвер» разработки «Лаборатории dk». Вы можете скачать его с сайта www.dklab.ru и установить на свой компьютер. В бесплатной ознакомительной версии системы эти средства включены в установочный комплект.

Клиентская часть системы требует только установки веб-браузера на компьютере пользователя (например, Internet Explorer 4.0 и выше или Netscape Navigator 4.0 и выше). Для использования стандартного встроенного HTML-редактора в ам понадобится браузер Internet Explorer 5.5 и выше. Однако в системе имеется альтернативный HTML-редактор FCKeditor, позволяющий полноценно работать под большинством современных браузеров (Internet Explorer, Mozilla, Netscape, FireFox, Opera). Во всех современных операционных системах браузеры входят в стандартную комплектацию системы.

Если вы используете систему на удаленном компьютере (например, на сервере хостинг-провайдера), то для закачки файлов по протоколу FTP или внесения изменений в файлы конфигурации Вам понадобится FTP-клиент – программа,

позволяющая работать с сервером по протоколу FTP. Если провайдер предоставляет терминальный (shell) доступ к серверу (например, по протоколам Telnet или SSH), в дальнейшей эксплуатации системы Вам также может понадобиться соответствующая программа для работы по этим протоколам (например, утилита Putty).

Процедура установки системы

Файловая система на поставляемом компакт-диске имеет следующую структуру:

папка **docs**		
файл	**developer.pdf**	Руководство разработчика
файл	**license.pdf**	Текст лицензионного соглашения
файл	**user.pdf**	Руководство пользователя
файл	**modules.pdf**	Сводное руководство по модулям
папка **files**		
папка	**images**	Папка для хранения картинок для сайта
папка	**install**	Папка с программой установки системы
папка	**netcat**	Папка с программными файлами
папка	**netcat_files**	Папка для хранения файлов, закачиваемых системой
файл	**.htaccess**	Файл с системными настройками
файл	**index.php**	Индексный программный файл системы
файл	**robots.txt**	Файл инструкций для поисковых роботов
файл	**readme.txt**	Краткая справка о системе и ее установке

Примечание: Все файлы на компакт-диске имеют кодировку UTF-8.

Для установки программных компонентов системы выполните следующие операции:

1. скопируйте все содержимое папки **files** с компакт-диска (4 папки и файлы - index.php, .htaccess и robots.txt) в домашний каталог сайта с сохранением структуры и регистра букв у названий файлов;

2. создайте базу данных для проекта или убедитесь, что она уже создана (подробнее см. ниже);

3. произведите системную настройку сайта (подробнее о процессе настройки см. ниже);

4. откройте в окне браузера URL **http://ДОМЕН_САЙТА/install/** (ДОМЕН_САЙТА – домен, по которому располагается сайт; для локальной версии сайта это может быть слово «localhost») и следуйте инструкциям;

5. после успешного завершения процесса установки сотрите папку **install**;

6. при желании или при выявлении ошибок дополнительно настройте конфигурационный файл **vars.inc.php** в папке **netcat** домашнего каталога сайта (подробнее о процессе настройки этого файла см. ниже).

Если Вы устанавливаете систему на хостинг-площадку (т.е. удаленный компьютер), Вам необходимо получить от хостинг-провайдера права доступа к своему аккаунту, в т.ч. доступ к управлению аккаунтом (обычно производится в браузере на сервере хостинг-провайдера), а также логин и пароль для соединения с сервером по протоколу FTP (при помощи FTP-клиентов осуществляется копирование файлов на сервер и редактирование конфигурационного файла).

Внимание! Если в папке, куда вы устанавливаете систему, уже есть файл index.php или какие-то из вышеуказанных папок, новые файлы будут записаны поверх старых; переименовывать файлы и папки системе категорически запрещено.

Создание базы данных

Если вы пользуетесь услугами хостинг-провайдера, скорее всего, у вас уже создана пустая база данных или есть возможность создать ее через веб-интерфейс управления Вашим аккаунтом. В этом случае Вам нужно создать базу данных – например, при помощи пакета **phpMyAdmin**, который часто входит в пакет программ на хостинг-площадке, и выяснить параметры доступа к ней (хост, имя базы, имя пользователя и пароль). Если при создании базы данных (БД) Вам нужно указать права пользователя к ней, укажите все возможные права. При возникновении проблем на этом этапе обратитесь к инструкциям хостинг-провайдера или к сотруднику его технической поддержки.

Если вы работаете под Windows, вы можете создать базу данных при помощи пакета **phpMyAdmin** или интерфейса СУБД MySQL.

Если вы работаете на нестандартной хостинг-площадке, вам следует обратиться к системному администратору или сотруднику службы поддержки хостинг-провайдера с просьбой помочь создать базу данных.

Настройка конфигурационного файла

В случае выявления ошибок или каких-то других помех, вы можете настроить конфигурационный файл **vars.inc.php** в папке **netcat** самостоятельно.

Отредактировать его можно, в частности, при помощи FTP-клиента. Ниже приведена таблица описания значения переменных. Каждая переменная определяется в соответствии с синтаксисом языка PHP:

```
$VAR_NAME = "Значение";
```

Переменная	Значение/описание	Пример
Параметры доступа к базе данных		
$MYSQL_HOST	Хост, с которого осуществляется доступ к базе данных. Если БД находится на том же сервере, что и система NetCat, хост чаще всего имеет значение "localhost".	"localhost"
$MYSQL_USER	Имя пользователя для доступа к базе данных.	"companyname"
$MYSQL_PASSWORD	Пароль пользователя базы данных.	"SuPeRpAsSwOrD"
$MYSQL_DB_NAME	Имя базы данных.	"companydb"
$MYSQL_ENCRYPT	Функция MYSQL, которая используется при шифровании паролей. Допустимые значения: MD5, SHA, PASSWORD, OLD_PASSWORD.	"MD5"
$MYSQL_TIMEZONE	Необязательный параметр. Устанавливает временную зону для базы данных (параметр TIME_ZONE). По существу, при указании этой переменной при загрузке системы выполнится запрос *SET TIME_ZONE = ВременнаяЗона*	"Europe/Moscow"
$SHOW_MYSQL_ERRORS	Показывать или не показывать MySQL ошибки на страницах проекта (front и back-office)"on"	"on"
$MYSQL_CHARSET	Кодировка соединения с БД	"utf8"
Параметры настроек авторизации		
$AUTHORIZE_BY	Поле в таблице пользователей, по которому происходит авторизация (по умолчанию – Логин). Чтобы использовать другие поля (e-mail, ID, имя…), их следует создать в разделе «Системные таблицы».	"Login"

$AUTHORIZATION_TYPE	Тип авторизации в интерфейсе системы администрирования: HTTP-авторизация (значение переменной "http"), Cookie-авторизация ("cookie"), Session - 32 разрядное хэш число, передаваемое в ссылке. Если PHP собран как CGI, то HTTP-авторизация недоступна.	"cookie"
Серверные настройки		
$PHP_TYPE	Тип сборки PHP. В большинстве случаев это модуль Apache (значение "module"), иногда – CGI (значение "cgi").	"module"
$REDIRECT_STATUS	Дает ли сервер возможность посылать браузеру заголовки содержания "header("Location: URL");". Возможные варианты: "on" (да), "off" (нет).	"on "
$ADMIN_LANGUAGE	Язык административной части NetCat "по-умолчанию". Если система по каким-то причинам не смогла определить язык интерфейса авторизованного пользователя, берется значение этой переменной.	"Russian"
$FILECHMOD	Права, проставляемые на файл, добавленный через веб-интерфейс системы.	0755
$DIRCHMOD	Права, проставляемые на директории для файлов, добавленных через веб-интерфейс системы.	0755
$ADMIN_AUTHTIME	Время жизни авторизации в секундах при $AUTHORIZATION_TYPE = session или cookie	86400
$ADMIN_AUTHTYPE	Время авторизации пользователя (только при типе авторизации cookie). Имеет три значения: 1.session - пользователь будет авторизован только на время данной сесии (до закрытия браузера) 2.always - пользователь будет авторизован в течение года 3.manual - под формой для ввода логина и пароля появится чекбокс, дающий возможность выбора времени авторизации пользователем. Если чекбокс выключен - авторизация будет действительна в течение данной сессии, если же чекбокс выбран - авторизация пользователя будет произведена на сутки.	"manual"

$NC_CHARSET	Корректная кодировка клиента для просмотра сайта (определяется сервером, если сервер – Russian Apache).	"utf-8"
$NC_UNICODE	Система работает с utf-8	1
$use_gzip_compression	Использовать сжатие страниц, выдаваемых сервером браузеру. По умолчанию выключено (false). Установите в true, чтобы включить сжатие.	false
$NC_REDIRECT_DISABLED	В случае установки этого параметра в 1, инструмент «Переадресация» будет отключён (позволяет снизить нагрузку на БД).	0
$NC_DEPRECATED_DISABLED	При установке этого параметра в 1 файл с «устаревшими» функциями не будет загружен. В некоторых дистрибутивах этого файла может не быть.	1
Настройки проекта		
$DOMAIN_NAME	Домен проекта. В случае если доступна серверная переменная $HTTP_HOST, можно установить значение $DOMAIN_NAME в $HTTP_HOST (по умолчанию).	"mydomain.ru"
$DOCUMENT_ROOT	Переменная окружения (устанавливается сервером), содержащая путь к папке, которая является корневым каталогом сайта (**обратите внимание, «слэша» на конце быть не должно**). Это поле нужно настраивать только в том случае, если модули/патчи устанавливаются не через веб-интерфейс.	"/usr/home/www"
$SUB_FOLDER	Подпапка в которой стоит NetCat	
$HTTP_DUMP_PATH	Папка для хранения дампов системы	"/netcat_dump/"
$HTTP_FILES_PATH	Папка для хранения файлов, загруженных через интерфейс системы	"/netcat_files/"
$HTTP_CACHE_PATH	Папка для файлового кэша	"/netcat_cache/"
Название разработчика		
$DEVELOPER_NAME	Имя разработчика, отображаемое на странице «О программе» (/netcat/admin/about/).	"NetCat"
$DEVELOPER_URL	Ссылка на веб-сайт разработчика.	"http://netcat.ru"
$DEVELOPER_EMAIL	Электронный адрес разработчика. Отображается на странице «О программе» и в правом нижнем углу страниц системы администрирования.	"support@netcat.ru"

Примечание: В большинстве случаев необходимо настроить только параметры доступа к базе данных.

Системная настройка сайта

Перед использованием (установкой) системы необходимо убедиться, что у файлов системы есть права на закачку файлов при помощи стандартных механизмов PHP на папки /**netcat_files, /netcat_dump. /netcat_cache** и **/netcat/tmp** без возможности исполнения закачанных файлов. Эти права можно установить, например, при помощи FTP-клиента (права 755 или 777). Для установки системы желательно также поставить аналогичные права на файл /**vars.inc.php**, чтобы в процессе установки в него автоматически записались все необходимые данные, в противном случае вам будет предложено сохранить его на диск для последующей закачки на сервер.

Обычно на этом системная настройка заканчивается, и можно переходить к следующему этапу установки. Если же в процессе установки или дальнейшей работы возникли проблемы, следует читать дальше.

Системная настройка осуществляется при помощи файла **.htaccess**, который поставляется вместе с системой. Однако некоторые хостинг-провайдеры не предоставляют такой услуги. Уточните в службе поддержки вашего провайдера наличие этой функции. В случае положительного ответа вам не придется осуществлять системную настройку – пропустите эту главу.

Стандартное содержание файл .htaccess:

```
DirectoryIndex index.php

AddDefaultCharset utf-8

# Если NetCat стоит в подпапке, например mysite, то
# ErrorDocument 404 /mysite/netcat/require/e404.php
# в противном случае
ErrorDocument 404 /netcat/require/e404.php

Options -Indexes
Options FollowSymLinks

<IfModule mod_php5.c>
php_flag magic_quotes_gpc on
php_flag display_errors on
php_value error_reporting 0
php_value arg_separator.output "&amp;"
```

```
    php_value mbstring.internal_encoding UTF-8
    </IfModule>



    <ifModule mod_rewrite.c>
    RewriteEngine On
    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteCond %{REQUEST_FILENAME} !-d
    RewriteCond %{REQUEST_FILENAME} !-l
    # Если NetCat стоит в подпапке, например mysite, то
    # RewriteRule ^(.+)$ /mysite/netcat/require/e404.php?
    REQUEST_URI=$1 [L,QSA]
    # в противном случае
    RewriteRule ^(.+)$ /netcat/require/e404.php?REQUEST_URI=$1
[L,QSA]
    RewriteRule .* - [E=HTTP_IF_MODIFIED_SINCE:%{HTTP:If-Modified-
Since}]
    RewriteRule .* - [E=HTTP_IF_NONE_MATCH:%{HTTP:If-None-Match}]
    </ifModule>
```

Если же управлять настройками сайта посредством .htaccess нельзя, вам необходимо выполнить несколько операций. Они могут быть доступны в интерфейсе управления вашим аккаунтом. При возникновении трудностей с настройкой обратитесь к сотруднику службы поддержки вашего провайдера. вам необходимо:

1. указать скрипт **/netcat/require/e404.php** как обработчик ошибки 404 для домена (если не используется модуль mod_rewrite);

2. корректно настроить mod_rewrite (инструкции для него описаны в файле .htaccess между <ifModule mod_rewrite.c> и </ifModule>).

После настройки системы вы можете приступать к работе с ней. Войдите в систему администрирования (вход находится по адресу http://ДОМЕН_САЙТА/netcat/admin/), используя логин и пароль установленные при инсталляции системы.

Внимание! Сразу же после установки в целях безопасности удалите директорию /install.

Настройка управления задачами (cron)

Управление задачами позволяет автоматически запускать нужные скрипты в необходимое время.

Запускать можно локальные скрипты, либо скрипты, находящиеся на любом другом хостинге.

Настройка:
Для настройки данной функции необходимо отредактировать файл: netcat/admin/crontab.php.

ВНИМАНИЕ! Вам следует установить верные значения:

- $DOCUMENT_ROOT - физический путь до папки, содержащей папку netcat (например, /var/httpd/example/www)
- $HTTP_HOST – домен, на котором работает сайт (без http://)

Затем через панель управления хостингом (или иным способом, в зависимости от вашего провайдера), нужно прописать в crontab-файле файл netcat/admin/crontab.php на исполнение каждую минуту.

Описание полей формы добавления или изменения задачи:

- минуты - запускать каждые n минут;
- часы - запускать каждые n часов;
- дни - запускать каждые n дней; **
- последний запуск - время и дата, когда в последний раз запускался скрипт;
- ссылка на скрипт - относительная или полная ссылка на скрипт, который необходимо выполнить.

Ваш тарифный план должен поддерживать выполнение cron.

***Если все три поля имеют значение 0, скрипт выполняться не будет; минимальный интервал 1 минута.*

Решение проблем

Если у вас возникли проблемы при установке системы, важно правильно классифицировать проблему. Если она относится к настройкам сервера хостинг-провайдера, обратитесь в его службу поддержки. Возможно, ее сотрудникам понадобится настоящее Руководство – вы можете переслать им аналогичный файл с компакт-диска. Наиболее часто встречающиеся проблемы и варианты их решения находятся в таблице ниже.

Проблема	Варианты решения
Не получается попасть в систему администрирования	Убедитесь, что в файле vars.inc.php правильно прописаны параметры для доступа к базе данных. Также возможна ситуация, когда Вы указываете тип авторизации «http», а PHP на Вашем сервере собран как CGI. Уточните этот вопрос у Вашего провайдера.
Ссылки с сайта вроде бы правильные, но сайт выдает 404 ошибку	Вы не настроили обработчик 404 ошибки. См. раздел «Системная настройка сайта».
Файлы через веб-интерфейс не закачиваются	Файлы PHP не имеют право записи в папку netcat_files. Установите права 755 или 777 на папку netcat_files (запись для всех без возможности выполнения для группы и остальных), либо попросите настроить эту возможность Вашего системного администратора.

Если же, по вашему мнению, проблема в другом, обратитесь к производителю системы, не забыв указать в письме ваш регистрационный номер и название компании. вы можете также попробовать решить проблему через сайт netcat.ru, на котором есть:

- форумы, посвященные решению различных проблем при использовании NetCat;
- контактная информация службы поддержки пользователей NetCat;
- разнообразные примеры, советы, рекомендации;
- различная документация к системе.

Предпроектная подготовка

Система NetCat позволяет оптимизировать и упростить процесс создания сайта и управления им в дальнейшем, однако перед началом создания сайта необходимо подготовить материалы для него. Ниже приведен примерный список материалов, которые рекомендуется подготовить до начала настройки NetCat:

1. Техническое задание (ТЗ)
2. Макеты дизайна всех типов страниц
3. Содержимое сайта (текстовая, графическая и прочая информация)

ТЗ должно описывать общую идею будущего сайта, а также следующие сущности в рамках сайта:

1. Структура сайта (рекомендуется в иерархическом виде, т.е. в виде дерева).

2. Описание компонентов и модулей: структура данных (список всех полей компонента), описание внешнего вида страниц каждого компонента (желательно не только в повествовательном, но и в схематическом виде).
3. Описание системы навигации по сайту (основных и дополнительных средств навигации).
4. Требования к дизайну сайта: рекомендуемая цветовая палитра, наличие/отсутствие большого объема графики, схематическое расположение различных блоков и пр. (если какие-либо страницы требуют нестандартного дизайна, они также должны быть описаны).
5. Описание титульной страницы (описание содержимого страницы, изменения в дизайне и пр.).
6. Требования к разграничению прав (опционально).

Создавать макеты дизайна желательно в соответствии с ТЗ. В частности, при создании макетов дизайна следует учитывать:

- структуру сайта (например, если количество разделов первого уровня велико и может «раздувать» страницу вширь);
- компоненты (перед созданием сложных компонентов желательно иметь дизайн-макет страницы данного компонента);
- требования к навигации (если в них указаны виды элементов навигации и требования к их оформлению);
- требования к дизайну.

Содержимое сайта – контент – также должно быть составлено и структурировано в соответствии с ТЗ. Собственно, процесс создания и структурирования контента может происходить параллельно с внедрением системы NetCat, т.к. обычно процесс наполнения сайта является последним или предпоследним (перед тестированием) этапом его создания.

Отметим, что эти пожелания носят лишь рекомендательный характер и применимо для создания сайтов на любых платформах.

Часть 4. Ввод и настройка структуры сайта

Создание сайтов

По умолчанию в системе присутствует один сайт, соответствующий домену www.DOMAIN.ZONE и DOMAIN.ZONE (например, www.netcat.ru), а также несколько начальных разделов. В стандартной поставке системы эти разделы введены для примера и могут быть удалены.

Войдите в настройки сайта и измените их. Значения полей подробно описаны в «Руководстве пользователя» (глава «Структура сайта»). В дальнейшем в системную таблицу «Сайт» можно добавить другие поля. Их настройка происходит таким же образом.

В большинстве случаев система должна содержать только один сайт. Вводить дополнительные сайты имеет смысл, например, в следующих случаях:

1. **Под управлением системы NetCat находится несколько сайтов**
   На одной копии системы NetCat можно создать несколько относительно независимых друг от друга сайтов. В этом случае в настройках каждого нового каталога/сайта необходимо указывать полный домен (www.example.ru) в поле «Ключевое слово». Однако в этом случае компоненты, списки, модули и пр. будут общими для всех сайтов.
2. **Сайт должен иметь некий глобальный раздел, предназначение которого существенно отличается от основного сайта**
   Пример: сайт, имеющий внутренний раздел для сотрудников или клиентов компании; совмещение традиционного сайта с extranet-системой. В этом случае необходимо настроить права доступа к новому сайту только для зарегистрированных или уполномоченных пользователей.
3. **Сайт имеет сложную структуру, которую имеет смысл разделить на несколько глобальных частей**
   Пример: по адресу www.example.ru располагается сайт группы компаний, а по адресам вида manufact.example.ru, sale.example.ru – сайты компаний, входящих в группу.

4. **На сайте должен быть сайт, выполняющий вспомогательные функции**
Пример: закрытый каталог, где в предварительном режиме выкладываются материалы, которые впоследствии должны быть опубликованы на основном сайте.

Создание разделов

После настройки сайта можно переходить к вводу всей его структуры. Начните добавлять разделы в сайт и подразделы в разделы. При добавлении раздела следует заполнить соответствующую форму, назначение полей которой описано в «Руководстве пользователя». В системную таблицу «Раздел» можно добавить другие поля. Их настройка происходит таким же образом.

Для добавления подразделов в разделы воспользуйтесь соответствующей ссылкой в меню работы с разделом или ссылкой в виде плюса в Карте сайта. Так вводится вся структура сайта.

Учтите, что в раздел нельзя добавить информацию, пока к нему не привязан хотя бы один из компонентов. Исключения составляют разделы, информация в которых выводится из других мест, например, при помощи функции nc_objects_list() из другого раздела или компонента раздела.

Прикрепление компонентов к разделам

Для каждого раздела (не считая символических) должен быть определен минимум один компонент. В том случае, если компонентов для раздела несколько, они могут выдаваться в виде закладок (оформленных любым образом – см. раздел «Макеты дизайна») или выводиться списком. Предположим, что все необходимые компоненты уже созданы (см. гл. «Компоненты»).

Добавить компонент в раздел можно при помощи закладки «Компоненты» на странице меню работы с разделом (см. «Руководство пользователя»).

Чаще всего в раздел достаточно добавить один компонент, после чего можно добавлять непосредственно информацию.

В силу особенностей NetCat, при прикреплении нескольких компонентов к разделу, все компоненты, начиная со второго по приоритету, будут работать в режиме «просмотр», даже если у них в действии по умолчанию указано другое. Поэтому, если вы хотите прикрепить несколько компонентов к разделу, один из которых — это форма добавления, то этот компонент должен иметь приоритет над остальными, что достигается указанием высшего значения в сооветствующем.

Примеры реализации нестандартных задач

В этом разделе приведены некоторые приемы управления структурой, которые можно реализовать при помощи системы NetCat.

1. **Сетевая структура сайта**
Сетевой тип структуры сайта отличается от иерархического тем, что каждый элемент структуры (раздел) может иметь более одного родительского раздела. Например, подраздел «Зарядные устройства для мобильных телефонов» (ключевое слово «zaryad») в каталоге товаров («catalog») должен относиться к разделам «Мобильные телефоны» («mobile») и «Аксессуары» («other»). Для реализации этой задачи нужно создать раздел «Зарядные устройства для мобильных телефонов» в каком-то одном из разделов, например, в «Мобильных телефонах». При этом относительный адрес раздела будет иметь вид /catalog/mobile/zaryad/, и раздел «Зарядные…» будет выводиться в списке подразделов раздела «Мобильные…». После этого необходимо создать символический раздел «Зарядные…» в разделе «Аксессуары», например, с таким же ключевым словом, указав «/catalog/mobile/zaryad/» как внешний URL. Таким образом, раздел «Зарядные…» также будет находиться в разделе «Аксессуары», но ссылаться будет на /catalog/mobile/zaryad/.

2. **Неоднотипный вывод пунктов меню одного уровня**
Пункты меню обычно выводятся по одному шаблону (точнее, как правило, используется один шаблон для активного пункта меню – выделение цветом или шрифтом – и один для неактивных пунктов). Тем не менее, при помощи системы NetCat можно организовать разнотипный вывод пунктов меню. Ниже приведено несколько примеров реализации таких задач.

a. **Выделение разных пунктов разными цветами**
Для реализации такого приема необходимо:

- добавить в системную таблицу «Разделы» новое поле, например, MenuColor, не обязательное, не наследуемое;
- установить значения этого поля для тех разделов, выделение которых другими цветами необходимо, например, «magenta», «#CCCCCC»;
- в макете установить цвет ссылок по умолчанию при помощи стандартных средств, например, таблицы стилей или атрибутов тега body;
- в настройках макета установить форматы вывода активного и неактивного элементов навигации с использованием этого поля, например, так:

```
$browse_sub[2]['active'] = "<a href='%URL'><span
color='%MenuColor'>%NAME</span></a>";
```

В этом случае для тех разделов, у которых указан цвет "red", будет выведен тег , а для тех, у которых поле пусто, тег , т.е. будет применен цвет по умолчанию.

b. **Некоторые пункты меню должны открываться в новом окне**
Реализуется аналогично предыдущему пункту: создается поле в таблице «Разделы» (например, Target), и в нужных разделах оно заполняется значением «_blank». В настройках макета формат вывода элементов навигации должен выглядеть примерно следующим образом:

```
$browse_sub[2]['active'] = "<a href='%URL'
target='%Target'>%NAME</a>";
```

c. **Каждому пункту может соответствовать свой значок («иконка»)**
В таблицах «Разделы» и «Сайты» создается поле (например, Icon) типа File, наследуемое. Для каждого раздела рисуется иконка и закачивается через стандартный интерфейс изменения настроек раздела. Целесообразно также закачать некоторые иконки по умолчанию через интерфейс изменения настроек сайта. Это следует сделать для тех случаев, когда раздел создан, а значка еще нет. Дальнейшие действия зависят от того, где именно применяется значок: в меню (например, рядом с каждой ссылкой на раздел первого уровня меню должен стоять значок) или в другой области раздела (т.е. на любой странице должен быть только один значок – соответствующий данному разделу). В первом случае изменяем настройки макета примерно так:

```
$browse_sub[2]['active'] = "<a href='%URL'><img src='%Icon'
width='10' height='10' alt='%NAME' border='0'> 
%NAME</a>";
```

Во втором случае нужно включить в нужное место страницы (футера или хедера) выражение примерно следующего содержания:

```
<img src=".$current_sub['Icon']." width='100' height='200'>
```

3. **Использование нескольких меню первого уровня**

Строго говоря, использовать несколько меню первого уровня нельзя, однако такую ситуацию можно сымитировать. Пример: одно меню, располагающееся по вертикали, должно содержать оглавление каталога продукции, а второе, горизонтальное – оглавление «корпоративного» раздела – вакансии, контакты, история и пр. Для решения этой задачи необходимо:

a. Ввести одну из этих двух веток структуры в отдельный раздел первого уровня (например, «Продукция»), а вторую (корпоративную) – оставить в виде разделов первого уровня. Таким образом, получится, что в списке разделов первого уровня, кроме «Вакансий», «Истории» и пр., появится раздел «Продукция».

b. В макете страницы (обычно, в хедере) указать как меню первого уровня (".s_browse_level(0, $browse_sub[0])."., где в массиве $browse_sub[0] хранится шаблон для вывода меню первого уровня), так и навигацию по разделу «Продукция» (".s_browse_sub(16, $browse_sub[2])."., где 16 – номер раздела «Продукция», а в массиве $browse_sub[2] хранится шаблон для вывода оглавления каталога товаров).

c. Выключить раздел «Продукция», чтобы он не показывался в «корпоративном» меню.

4. **Дублирование меню любого уровня (обычно первого) внизу страницы в другом оформлении**

Эта задача реализуется путем вызова функции s_browse_level() два раза – в футере и хедере страницы:

-В хедере: ".s_browse_level(0, $browse_sub[0])."., где в массиве $browse_sub[0] хранится шаблон вывода меню для хедера;

-В футере: ".s_browse_level(0, $browse_sub[1])." , где в массиве $browse_sub[1] хранится шаблон вывода меню для футера.

5. Меню сайта в виде выпадающих списков (тег <select>)

Для реализации данного вида меню нужно только соответствующим образом настроить вывод меню (например, первого уровня) в настройках макета:

```
$browse_sub['prefix'] =  "<form action='' name='navigat'
method='get'
onsubmit='this.document.location.href=this.navigation.value;
return false;'><select size='1' name='navigation'>";

$browse_sub['suffix'] =  "</select><input type='submit'
value='Перейти'></form>";

$browse_sub['active'] = "<option value='%URL' selected>
%NAME</option>";

$browse_sub['unactive'] = "<option value='%URL'> %NAME</option>";

$browse_sub['divider'] = "";
```

6. Создание «мастеров», экскурсий и пр. (функционалов, требующих последовательного перехода по некоторым разделам)

Для реализации мастера такого вида следует:

-добавить в таблицу «Раздел» ненаследуемое поле (например, NextLink);

-в нужном месте макета (хедера или футера) внести примерно следующее выражение:

```
".opt($current_sub['NextLink'], "<a href='".
$current_sub['NextLink']."'>Далее</a>")."
```

В приведенном примере ссылка «Далее» будет появляться только в тех разделах, в которых это поле заполнено.

Часть 5. Дизайн сайта

Подготовка макетов страниц

Каждый макет дизайна, используемый на сайте, предварительно должен быть создан в виде файла-макета в формате HTML. Изображения, присутствующие в оформлении страницы, желательно размещать в одной папке или системе папок, которая будет затем скопирована на сервер.

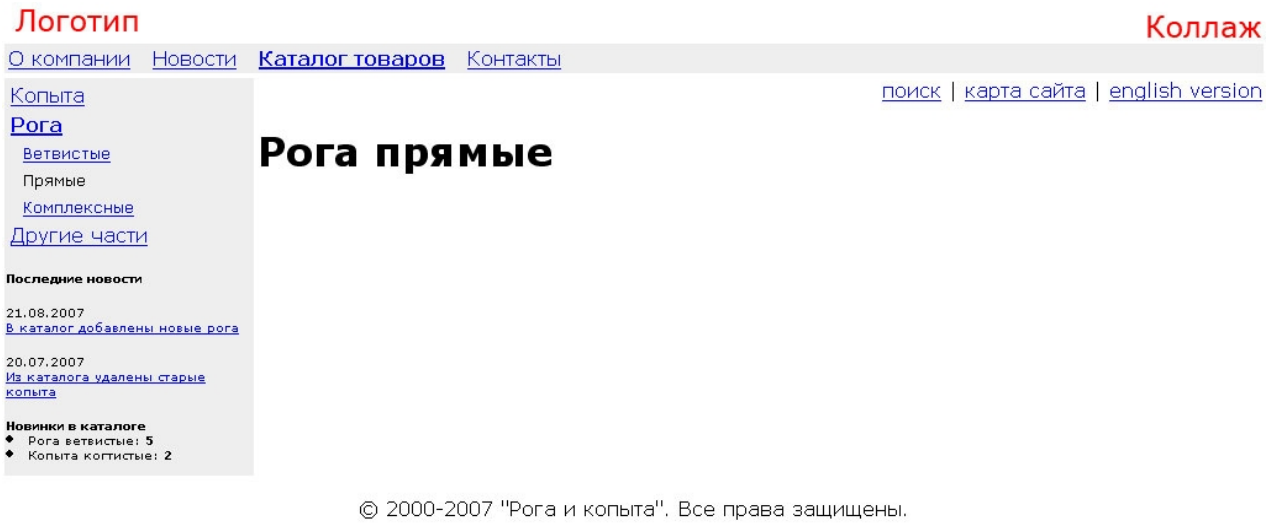
В макете внутренних страниц желательно учесть детали, приведенные ниже:

1. Способ выделения активных разделов. Например, текущий раздел первого уровня должен выделяться цветом фона и текста, а текущий раздел второго уровня – отсутствием ссылки на нем. Макет должен содержать все возможные блоки навигации (1-2-3… уровня, «хлебные крошки» и пр.) и в каждом блоке должны присутствовать как активные, так и неактивные элементы.
2. Внешний вид заголовка страницы (тег <title>). Заголовок страницы может иметь вид, например, «Название компании», «Название текущего раздела», «Компания / Раздел 1 / Раздел 11» и пр.
3. Если какой-либо элемент оформления или навигации присутствует не на всех страницах, необходимо создать макет таким образом, чтобы при отсутствии данного элемента страница не исказилась, либо, если это невозможно, создать и использовать разные макеты для данных ситуаций.
4. Таблицу стилей можно иметь как в теле макета, так и во внешнем файле. Использование двойных кавычек в коде макета желательно минимизировать, заменяя их на апострофы. Если же кавычки необходимы, то потребуется их экранирование – перед каждой кавычкой необходимо поставить обратный слэш \. В разделе «Инструменты» имеется функционал экранирования спецсимволов. Он автоматически заменяет все спецсимволы во введенном коде на безопасную комбинацию. Это связано с тем, что кавычки " являются спецсимволом PHP. Например, ими мы выделяем функцию: ".s_browse_level(0, $browse_sub[0])." . Для упрощения работы мы рекомендуем Вам использовать одинарные кавычки ' (они равноправны двойным практически во всех стандартах, включая HTML, XHTML).

Кроме того:

- необходимо четко разграничить все три части страницы: хедер, содержательную часть и футер;
- если для навигации используются графические изображения (названия разделов или иконки), следует нарисовать картинки для всех разделов;
- если для оформления разных разделов используются разные стили или коллажи, они также должны быть созданы.

Создадим простейший HTML-макет и поэтапно опишем процесс конвертации его в макет NetCat. Пусть необходимо, чтобы страницы сайта выводились в таком макете:



Приведем HTML-текст такого макета:

```
<html>

<head>
<title>Моя компания / Каталог товаров / Рога / Прямые</title>
<style type=text/css>
body {background-color:white; margin:0; font-family:verdana;}
</style>
</head>

<body>
<table width='100%' cellpadding='3'>
<tr>
<td><img src='/images/logo.gif' alt=''></td>
<td align='right' valign='bottom'><img
src='/images/collage_12.gif'></td>
</tr>
```

```
<tr>
<td colspan='2' bgcolor='#EEEEEE'><a href='/about/'>О
компании</a>   <a href='/news/'>Новости</a>   <b><a
href='/catalog/'>Каталог товаров</a></b>   <a
href='/contacts/'>Контакты</a></td>
</table>

<table width='100%'>
<tr>
<td valign='top' width='200' bgcolor='#EEEEEE'>
<p>
<table width=100% cellpadding=2>
<tr><td><a href=/roga/>Копыта</a></td></tr>
<tr><td><b><a href=/roga/>Рога</a></b></td></tr>
<tr><td><font size=-1>  <a
href=/roga/vetvistye/>Ветвистые</a></td></tr>
<tr><td><font size=-1>  Прямые</td></tr>
<tr><td><font size=-1>  <a
href=/roga/complexnye/>Комплексные</a></td></tr>
<tr><td><a href=/roga/>Другие части</a></td></tr>
</table>
<p>
<font size=-2>
<b>Последние новости</b>
<p>
21.08.2007<br>
<a href=/news/news_12.html>В каталог добавлены новые рога</a>
<p>
20.07.2007<br>
<a href=/news/news_11.html>Из каталога удалены старые
копыта</a>
<p>
<b>Новинки в каталоге</b>
<li> Рога ветвистые: <b>5</b><br>
<li> Копыта когтистые: <b>2</b><br>
<p>
</font>
</td>

<td valign='top'>
<div align=right><a href=/search/>поиск</a> | <a
href=/map/>карта сайта</a> | <a href=/english/>english
version</a></div>
<h1>Рога прямые</h1>
<p>
</td>
</tr>
</table>
<p align='center'>
&copy; 2000-2007 "Рога и копыта". Все права защищены.
```

```
</body>
</html>
```

Очевидно, что эта страница раздела «Прямые рога» подраздела «Рога» каталога товаров. Выделим блоки, которые необходимо формировать динамически. Это:

- заголовок страницы (тег title);
- коллаж справа вверху (для каждого раздела он должен быть уникальным);
- меню первого уровня («О компании», «Новости» и пр.);
- меню второго уровня («Копыта», «Рога» и пр.);
- меню третьего уровня («Ветвистые», «Прямые» и пр.) – причем, меню выводится под родительским меню, и только для активного раздела;
- блок последних новостей;
- количество новинок;
- заголовок страницы.

Для удобства мы выделили бы еще три динамических блока:

- блок стилей CSS;
- вспомогательное меню «Поиск», «Карта сайта» и пр.;
- текущий год в блоке копирайтов.

Выделим динамические блоки жирным шрифтом и снабдим комментариями:

```
<html>

<head>
<title>Моя компания / Каталог товаров / Рога / Прямые</title>
<!-- в заголовке title нужно выводить путь до текущей
страницы, т.е. «хлебные крошки» -->
<style type=text/css>
body {background-color:white; margin:0; font-family:verdana;}
</style>
</head>

<body>
<table width='100%' cellpadding='3'>
<tr>
<td><img src='/images/logo.gif'></td>
<td align='right' valign='bottom'><img
src='/images/collage_12.gif'></td> <!-- динамический коллаж в
зависимости от раздела -->
</tr>
<tr>
```

```
<td colspan='2' bgcolor='#EEEEEE'><a href='/about/'>О
компании</a>   <a href='/news/'>Новости</a>   <b><a
href='/catalog/'>Каталог товаров</a></b>   <a
href='/contacts/'>Контакты</a></td> <!-- меню первого уровня
-->
</table>

<table width='100%'>
<tr>
<td valign='top' width='200' bgcolor='#EEEEEE'>
<p>
<table width=100% cellpadding=2>
<tr><td><a href=/roga/>Копыта</a></td></tr>
<tr><td><b><a href=/roga/>Рога</a></b></td></tr>
<tr><td><font size=-1>  <a
href=/roga/vetvistye/>Ветвистые</a></font></td></tr>
<tr><td><font size=-1>  Прямые</font></td></tr>
<tr><td><font size=-1>  <a
href=/roga/complexnye/>Комплексные</a></font></td></tr>
<tr><td><a href=/roga/>Другие части</a></td></tr>
</table> <!-- меню второго и третьего уровней -->
<p>
<font size=-2>
<b>Последние новости</b>
<p>
21.08.2007<br>
<a href=/news/news_12.html>В каталог добавлены новые рога</a>
<p>
20.07.2007<br>
<a href=/news/news_11.html>Из каталога удалены старые
копыта</a>
<p> <!-- выборка двух последних объектов из раздела «новости»
-->
<b>Новинки в каталоге</b>
<li> Рога ветвистые: <b>5</b><br>
<li> Копыта когтистые: <b>2</b><br> <!-- выборка количества
объектов, удовлетворящих некоему условию «последние» -->
<p>
</font>
</td>

<td valign='top'>
<div align=right><a href=/search/>поиск</a> | <a
href=/map/>карта сайта</a> | <a href=/english/>english
version</a></div> <!-- вспомогательное меню -->
<h1>Рога прямые</h1> <!-- заголовок страницы -->
<p>
</td>
</tr>
</table>
```

```
<p align='center'>
&copy; 2000-2007 "Рога и копыта". Все права защищены. <!--
текущий год -->
</body>
</html>
```

Ниже приведем последовательность действий, необходимых для конвертации этого файла в формат макета NetCat.

Конвертация и ввод макетов страниц

Итак, макет создан. Следующий шаг – конвертация его в формат NetCat. Первый этап – копирование всех необходимых файлов (картинок, flash-роликов, внешних подключаемых файлов и пр.) на сервер. Обычно их копируют при помощи FTP-клиента в каталог images. В исходном файле в процессе конвертации следует заменить адреса картинок на новые. Рекомендуется использовать относительные ссылки – «/images/logo.gif» вместо «http://www.example.ru/images/logo.gif» и «images/logo.gif» – особенно если сайт разрабатывается по временному адресу.

Следующий шаг – разбиение макета на две части – хедер и футер. В примере, приведенном в предыдущем пункте, «разрыв» должен находиться после заголовка страницы:

```
<h1>Рога прямые</h1>
<p>
```

Следующий шаг – непосредственно конвертация макета. В обеих частях макета необходимо заменить динамические элементы специальными выражениями.

Разберем подробно каждый пункт. Прежде всего, следует учитывать, что для системы NetCat содержимое хедера и футера является обычным строковым выражением, которое нужно отобразить, выполнив прежде функцию PHP **eval**().
Т.е. хедер следует воспринимать в контексте следующего оператора:

```
eval("echo \"".$header."\";").
```

Это значит, что для хедера (и футера) действуют все правила представления переменных, а именно:

-при вызове функций необходимо закрывать кавычку, сцеплять строку с функцией при помощи точки, сцеплять функцию со следующей строкой,

открывать кавычку, например: «…text before".func($parameters)."text after…»;
-символы обратного слеша «\», кавычки «"», знак доллара «$» нужно маскировать при помощи обратного слеша, т.е. для отображения символа «\» нужно писать «\\», а для отображения кавычки – «\"» (для автоматической маскировки специальных символов вы можете воспользоваться соответствующим функционалом системы);
-переменные сцеплять со строкой необязательно: «…some text $f_some_var some another text» или «text ${f_some_var} some text»;
-можно использовать все специальные символы – «\n» (перенос строки), «\t» (табуляция) и пр.; также доступны функции PHP и специальные функции NetCat (полный перечень см. в Приложении 2).

Итак, заменяем динамические элементы необходимыми переменными и функциями. Более детальное описание указанных ниже функций и переменных вы можете найти в Приложении 2 этого Руководства.

1. **Содержимое тега <title>**
В нашем примере в заголовке окна браузера выводится путь до текущей страницы. Полный путь (с тегами) выводится при помощи функции s_browse_path($array), где $array – массив, содержащий шаблон вывода пути до текущей страницы); для вывода его без тегов следует вызвать функции очищения текста от тегов:

```
<title>".strip_tags(s_browse_path($browse_top))."</title>
```

Для форматирования вывода элемента навигации в данном случае использован массив $browse_top, который определяется в настройках макета:

```
$browse_top['active_link'] = "%NAME";
$browse_top['unactive'] = "%NAME";
$browse_top['active'] = "%NAME";
$browse_top['divider'] = " / ";
```

2. **Содержимое таблицы CSS**
Для того чтобы таблица CSS была единой для всех макетов, были произведены следующие действия:
1. в системную таблицу «Макеты дизайна» (меню «Инструменты – Системные таблицы») было добавлено текстовое поле «Таблица стилей» (английское название поля – «CSS»);
2. в корневом (стандартном) макете это поле было заполнено;

3. в хедере между тегами <style type=text/css> и </style> была добавлена макропеременная %CSS.

3. Вспомогательное меню «Поиск», «Карта сайта», «English version»

Теоретически данный блок можно прописать в макете «жестко», т.е. HTML-текстом. Но вполне возможно, что через некоторое время нужно будет убрать или добавить какой-либо пункт, изменить название. Чтобы для внесения этих изменений не нужно было менять код всех макетов, где присутствует блок вспомогательной навигации, было сделано следующее:

1. в структуру сайта был добавлен выключенный раздел «Вспомогательное меню» (его номер, например, 113);
2. все три пункта вспомогательного меню были добавлены в этот раздел в качестве подразделов;
3. в настройки макета добавлен массив $browse_sub_help, который описывает формат вывода данного элемента навигации. Разберем его подробно:

```
$browse_sub_help['prefix'] = "";

$browse_sub_help['suffix'] = "";

// Префикс и суффикс отсутствуют

$browse_sub_help['unactive'] = "<a href=%URL class=menu>
%NAME</a>";

$browse_sub_help['active'] = "<a href=%URL class=menu>
%NAME</a>";

$browse_sub_help['active_link'] = "<a href=%URL
class=menu>%NAME</a>";

// Формат вывода самый простой

$browse_sub_help['divider'] = " | ";

// В качестве разделителя между элементами меню
используется прямая черта, окруженная пробелами.
```

4. в макет вставлена функция отображения списка подразделов 113-го раздела:

```
".s_browse_sub(113,$browse_sub_help)."
```

4. Коллаж для текущего раздела

Согласно дизайнерскому решению, на каждой странице сайта отображается коллаж. Причем, нужно понимать, что, если по какой-то причине для некоего раздела коллаж не был создан, это не должно привести к «битым» картинкам. Для реализации этого решения приведем желательную последовательность действий:

1. в системную таблицу «Разделы» добавляем поле «Collage» (Коллаж) типа «Файл». В свойствах поля указываем, что оно наследуемое (т.е. для всех подразделов внутри «Каталога товаров» это поле будет наследоваться из раздела «Каталога товаров», если на каком-нибудь уровне оно не будет переопределено);
2. для каждого раздела первого уровня создаем коллаж;
3. в форме редактирования настроек всех разделов закачиваем нужный коллаж;
4. в нужном месте макета дизайна указываем его следующим образом:

```
<img src='".$current_sub['Collage']."'>
```

5. Меню первого уровня

В нашем примере формат меню первого уровня практически идентичен формату вспомогательного меню. В тексте макета вместо списка ссылок на разделы надо вызвать функцию:

```
".s_browse_level(0, $browse_sub[0])."
```

А в настройках макета определить формат отображения пунктов этого меню:

```
$browse_sub[0] = "";

$browse_sub[0] = "";

// Префикс и суффикс отсутствуют

$browse_sub[0] = "<a href=%URL class=menu>%NAME</a>";

$browse_sub[0] = "<b><a href=%URL class=menu>
%NAME</a></b>";

// Активный раздел выделим жирным

$browse_sub[0] = "<b>%NAME</b>";

// Если посетитель находится на странице, куда ведет
ссылка из меню, то сама эта ссылка, очевидно, не нужна

$browse_sub[0] = "   ";
```

6. Меню второго уровня

Меню первого уровня функционально сложнее первого уровня. В данном случае необходимо вывести активный пункт меню «раскрытым», т.е. под активным пунктом меню выводится список всех его подменю (меню третьего уровня). Для реализации этой задачи использовался стандартный механизм настроек элемента навигации:

1. В настройки макета добавляем массив $browse_sub[1]:

```
$browse_sub[1]['prefix'] = "\";
```

```
global \$browse_sub;

\$result.=\"<table width=100% cellpadding=2>";

// Настройки раздела обрабатываются функцией eval() языка
PHP. При выполнении этой функции в данном случае массив
$browse_sub не будет «виден» внутри цикла, а этот массив
нам понадобится для вывода меню второго уровня. Поэтому в
префиксе текст «прерывается» при помощи такого выражения:
«"\";...\$result.=\"», а вместо троеточия вставляется
объявление глобальной переменной – массива $browse_sub.
Такая конструкция нужна только для данного примера – если
бы был использован стандартный принцип вывода навигации
(когда на любой внутренней странице видно только меню
первого уровня и меню второго уровня, относящееся к этому
разделу), «разрывать» текст не было бы необходимости. После
разрыва идет начало таблицы, в которой будет выведено меню.

$browse_sub[1]['suffix'] = "</table>";

// Суффикс меню второго уровня – таблица закрывается.

$browse_sub[1]['active'] = "<tr><td ><b><a href=%URL>
%NAME</a></b></td></tr>".s_browse_level(2,$browse_sub[2]);

// Формат вывода активного элемента меню первого уровня.
После названия со ссылкой выводится меню третьего уровня –
функция s_browse_level(2,$browse_sub[2]), последний
параметр которой нужно будет описать – он отвечает за
форматирование вывода меню третьего уровня (см. ниже).

$browse_sub[1]['active_link'] = "<tr><td><b>
%NAME</b></td></tr>".s_browse_level(2,$browse_sub[2]);

// После активного элемента меню необходимо вывести все его
подразделы в том шаблоне, который был объявлен глобальным в
prefix.

$browse_sub[1]['unactive'] = "<tr><td ><a href=%URL>
%NAME</a></td></tr>";;

$browse_sub[1]['divider'] = "";

// Разделитель не используется.
```

2. В настройках раздела определяем внешний вид меню второго уровня:

```
$browse_sub[2]['prefix'] = "";

$browse_sub[2]['suffix'] = "";

$browse_sub[2]['active'] = "<tr><td><font size=-1>  <a
href=%URL>%NAME</a></font></td></tr>";

$browse_sub[2]['active_link'] = "<tr><td><font size=-
1>  %NAME</font></td></tr>";
```

```
$browse_sub[2]['unactive'] = $browse_sub[2][active];
$browse_sub[2]['divider'] = "";
```

3. В нужное место в макете вставляем вызов функции, отвечающей за вывод меню второго уровня:

```
".s_browse_level(1,$browse_sub[1])."
```

7. Заголовок страницы

Стандартный способ вывести заголовок страницы (название текущего раздела) – написать переменную $f_title. Но данный случай, в силу своей специфики, не подходит под стандартный способ, потому что, как видно из меню третьего уровня, раздел называется «Прямые», а необходимо выдать полное название раздела. Для этого нам неоходимо:

1. создать в системной таблице «Разделы» поле «Полное название» (название по-английски, например, FullName);
2. заполнить его в нужных разделах;
3. в макете вывести его значение:

```
".$current_sub['FullName']."
```

Однако, если это поле в каком-то разделе заполнено не будет, заголовок не выведется. Поэтому целесообразно проверять его наличие и в случае его отсутствия выводить стандартное название раздела:

```
".opt_case($current_sub['FullName'], $current_sub['FullName'],
$f_title)."
```

8. Выборка двух последних новостей

Это, пожалуй, одна из самых простых операций. Например, номера раздела «Новости» - 12, а номер компонента раздела – 30. Тогда вывести последние два объекта новостей можно будет вызовом функции:

```
".s_list_class(12,30,"recNum=2")."
```

Забегая немного вперед, стоит указать, что формат вывода новостей во вставке оформления страницы может отличаться от списка новостей в соответствующем разделе. В этом случае альтернативный формат должен быть описан в шаблоне вывода компонента «Новости». Поэтому следует подать дополнительный параметр на функцию s_list_class(), который будет обработан в тексте шаблона компонента:

```
".s_list_class(12,30,"recNum=2&tmpl=short")."
```

9. Количество новинок в каталоге

Среди стандартных функций NetCat нет функции, которая бы выводила количество товаров в каком-либо разделе с какими-либо условиями выборки, поэтому данную выборку придется формировать прямым запросом к базе. Предположим, под новыми товарами мы подразумеваем товары, добавленные сегодня. Компонент «Товары» имеет номер 57, т.е. его экземпляры хранятся в таблице Message57. SQL-запрос к базе в этом случае будет выглядеть так:

```
SELECT a.Subdivision_Name AS name, b.Sub_Class_ID AS id,
count(c.Message_ID) AS cnt FROM Subdivision AS a, Sub_Class AS
b, Message57 AS c where a.Subdivision_ID=b.Subdivision_ID AND
b.Sub_Class_ID=c.Sub_Class_ID AND DATE(c.Created)=CURDATE()
GROUP BY name ORDER BY name
```

Отформатируем результаты запроса при помощи соответствующей функции NetCat:

```
".listQuery("select a.Subdivision_Name as name, b.Sub_Class_ID
as id, count(c.Message_ID) as cnt from Subdivision as a,
Sub_Class as b, Message57 as c where
a.Subdivision_ID=b.Subdivision_ID and
b.Sub_Class_ID=c.Sub_Class_ID and DATE(c.Created)=CURDATE()
group by name order by name", "<li>\$data[name]: <b>\
$data[cnt]</b>")."
```

10. Содержательная часть страницы

Для определения места вывода содержательной части страницы не используются никакие вставки. Это место в макете является разрывом между хедером и футером. Т.е. текст до этого места записывается в хедер, а после – в футер.

11. Текущий год

Текущий год вставляется простейшей функцией PHP:

```
&copy; 2000-".date('Y')."
```

Таким образом, мы получили следующие значения полей для макета:

Шаблоны вывода навигации (настройки макета):

```
$browse_top['active_link'] = "%NAME";
$browse_top['unactive'] = "%NAME";
$browse_top['active'] = "%NAME";
$browse_top['divider'] = " / ";

$browse_sub_help['prefix'] = "";
```

```
$browse_sub_help['suffix'] = "";
$browse_sub_help['unactive'] = "<a href=%URL class=menu>
%NAME</a>";
$browse_sub_help['active'] = "<a href=%URL class=menu>
%NAME</a>";
$browse_sub_help['active_link'] = "<a href=%URL class=menu>
%NAME</a>";
$browse_sub_help['divider'] = " | ";

$browse_sub_help[0] = "";
$browse_sub_help[0] = "";
$browse_sub_help[0] = "<a href=%URL class=menu>%NAME</a>";
$browse_sub_help[0] = "<b><a href=%URL class=menu>
%NAME</a></b>";
$browse_sub_help[0] = "<b>%NAME</b>";
$browse_sub_help[0] = "   ";

$browse_sub[2]['prefix'] = "";
$browse_sub[2]['suffix'] = "";
$browse_sub[2]['active'] = "<tr><td><font size=-1>  <a
href=%URL>%NAME</a></font></td></tr>";
$browse_sub[2]['active_link'] = "<tr><td><font size=-1> 
%NAME</font></td></tr>";
$browse_sub[2]['unactive'] = $browse_sub[2]['active'];
$browse_sub[2]['divider'] = "";
$browse_sub[1]['prefix'] = "\";global \$browse_sub;\
$result.=\"<table width=100% cellpadding=2>";
$browse_sub[1]['suffix'] = "</table>";
$browse_sub[1]['active'] = "<tr><td ><b><a href=%URL>
%NAME</a></b></td></tr>".s_browse_level(2,$browse_sub[2]);
$browse_sub[1]['active_link'] = "<tr><td><b>
%NAME</b></td></tr>".s_browse_level(2,$browse_sub[2]);
$browse_sub[1]['unactive'] = "<tr><td ><a href=%URL>
%NAME</a></td></tr>";
$browse_sub[1]['divider'] = "";
```

Хедер:

```
<html>

<head>
<title>".strip_tags(s_browse_path($browse_top))."</title>
<style type=text/css>
%CSS
</style>
</head>

<body>
<table width='100%' cellpadding='3'>
<tr>
<td><img src='/images/logo.gif'></td>
<td align='right' valign='bottom'><img src='".
$current_sub[Collage]."'></td>
</tr>
<tr>
<td colspan='2' bgcolor='#EEEEEE'>".s_browse_level(0,
$browse_sub[0])."</td>
</table>

<table width='100%'>
<tr>
<td valign='top' width='200' bgcolor='#EEEEEE'>
<p>
".s_browse_level(1, $browse_sub[1])."
<p>
<font size=-2>
<b>Последние новости</b>
<p>
".s_list_class(12,30,"recNum=2&tmpl=short")."
<b>Новинки в каталоге</b>
".listQuery("select a.Subdivision_Name as name, b.Sub_Class_ID
as id, count(c.Message_ID) as cnt from Subdivision as a,
Sub_Class as b, Message57 as c where
a.Subdivision_ID=b.Subdivision_ID and
b.Sub_Class_ID=c.Sub_Class_ID and DATE(c.Created)=CURDATE()
group by name order by name", "<li>\$data[name]: <b>\
$data[cnt]</b>")."
<p>
</font>
</td>

<td valign='top'>
<div align=right>".s_browse_sub(113,$browse_sub_help)."</div>
".opt_case($current_sub['FullName'], $current_sub['FullName'],
$f_title)."<p>
```

Футер:

```
</td>
</tr>
</table>
<p align='center'>
&copy; 2000-".date('Y')." \"Рога и копыта\". Все права
защищены.
</body>
</html>
```

Таблица стилей:

```
body {background-color:white; margin:0; font-family:verdana;}
```

Обратите внимание, что:

1. В настройках макета массив $browse_sub[2] определен выше, чем $s_browse_sub[1]. Это необходимо, чтобы $browse_sub[2] был «виден» в $browse_sub[1].
2. Атрибуты HTML-тегов сознательно указаны то в одинарных кавычках, то вообще без кавычек – двойные кавычки не использованы. Это связано с тем, что двойная кавычка является признаком конца/начала строки в языке PHP. Все двойные кавычки необходимо маскировать обратным слэшем, как это сделано в футере:

```
\"Рога и копыта\"
```

Другие функции и переменные, доступные для использования в макетах, см. в Приложении 2.

После добавления макета выберите его в настройках сайта (или нужных разделов). Номер макета также можно напрямую подавать на страницу как параметр. Это целесообразно, когда макет нужно тестировать на рабочем сайте. Например, если есть задача протестировать внешний вид страницы «www.example.ru/about/» с новым макетом (например, номер 3), сделав процесс тестирования незаметным для посетителей сайта, нужно вызывать страницу таким образом: «www.example.ru/about/?template=3».

Аналогично решается задача создания версии для печати:

1. создается (и тестируется) новый облегченный макет, предназначенный для распечатки;

2.
3. в тексте макета (если ссылка «версия для печати» нужна на всех страницах, использующих макет) или компонента (если ссылка нужна только на страницах некоторых компонентов) ввести примерно следующий текст:

```
<a href='?template=3'>версия для печати</a>
```

Использование дополнительных полей

Дополнительные поля можно использовать как для удобства будущей правки макета, так и для сокращения трудозатрат при использовании нескольких макетов, незначительно отличающихся друг от друга.

Что касается первого случая, то в нашем примере удобно некоторые части кода вынести в отдельные поля: ключевые слова, описание страницы, CSS-таблицу. Для вынесения в отдельное поле, например, таблицы стилей, добавим новое поле CSS в системную таблицу «Макеты страниц». Поле должно быть типа Текстовый блок. Перенесем в него содержимое CSS-таблицы, а в хедер внесем название поля со знаком процента:

```
<style type='text/css'>
<!--
%CSS
 -->
</style>
```

Использовать дополнительные поля для сокращения трудозатрат при нескольких макетах можно при помощи механизма наследования. Например, на сайте нужно использовать два макета, отличающиеся каким-либо фрагментом. Для реализации этой задачи нужно:

1. добавить в таблицу «Макеты страниц» поле, например, SomeField (тип Текстовый блок);
2. создать первый макет;
3. перенести HTML-код варьируемой области в это поле, заменив его в хедере/футере выражением %SomeField;
4. добавить новый макет, дочерний по отношению к основному (для этого нужно нажать на значок «плюс» рядом с названием родительского макета);
5. заполнить в новом макете только поле SomeField, указав в нем тот фрагмент кода, который нужно выводить на страницах данного макета;

6. указать макеты в настройках тех разделов, где они нужны.

В любом «подмакете» дизайна доступно выражение %Header и %Footer, содержащие значения, соответственно, начала и конца страницы родительского макета.

Шаблоны вывода навигации

В каждом макете дизайна есть поле с названием «Шаблоны вывода навигации» (настройки макета). В нем содержатся массивы данных. Например, для вывода навигации на сайте используется s_browse_level(), вторым параметром у которой как раз и указан один из массивов, в котором содержится оформление меню в виде HTML.

Шаблоны используются в таких функциях, как: s_browse_level(), s_browse_cc(), s_browse_sub(), s_browse_catalogue(), s_browse_path_range(), s_browse_path(), browse_messages(). Описание всех функций вы можете найти в Приложении 2 данного руководства.

Рассмотрим пример:

```
$browse_sub[0]['prefix'] = "<ul>";
$browse_sub[0]['suffix'] = "</ul>";
$browse_sub[0]['active'] = "<li><b><a href=%URL>
%NAME</b></a>".s_browse_level(1,$browse_sub[1])."</li>";
$browse_sub[0]['active_link'] = "<li><b>
%NAME</b>".s_browse_level(1,$browse_sub[1])."</li>";
$browse_sub[0]['unactive'] = "<li><a href=%URL>%NAME</a></li>";
$browse_sub[0]['divider'] = "";
```

Элементы указывают:

- prefix – предшествует списку выводимых объектов. Например, в нем можно открыть таблицу.
- suffix – идет после списка выводимых объектов. Например, в нем можно закрыть таблицу.
- active – выводит активный элемент, одна его ссылка (%URL) не совпадает с адресом текущей страницы. Например, мы находимся в полном выводе «Новости».
- active_link – выводит активный элемент, и его ссылка (%URL) совпадает с адресом текущей страницы. Например, мы находимся на странице списка новостей.
- unactive – выводит неактивный элемент.

- divider – разделитель между элементами списка навигации, например, Раздел 1 / Раздел 2. / - это разделитель.
- sortby – позволяет сортировать выводы списка в нужном порядке. Чаще всего этот элемент не указывается, тогда список сортируется по приоритету (Priority).
- nocache — флаг, позволяющий запретить кэширование (при наличии модуля «Кэширование»). Подробнее смотрите в «Руководстве по модулям».

Функция s_browse_level(1,$browse_sub[1]) в элементах active и active_link выводит следующий уровень в текущем разделе, т.е. список его подразделов. Это условие выполняется только для 1-го и второго уровня вложенности.

В шаблонах вывода навигации есть ряд макропеременных:

- %NAME – название элемента (раздела);
- %PARENT_SUB – номер родительского раздела (только для разделов);
- %KEYWORD – ключевое слово раздела (только для разделов);
- %SUB – номер раздела (только для разделов);
- %COUNTER – номер выводимого элемента в списке (начиная с нуля).

Если вам не хватает данных макропеременных, вы можете использовать внутренний массив $data[]. Он содержит все данные из таблицы Subdivision о текущей выводимой функции в разделе.
Например:

```
$browse_sub[0]['unactive'] = "<li>
<a href=\".\$data[\$i][Hidden_URL].\">\".\$data[\$i]
[Subdivision_Name].\"</a></li>";
```

равносильно записи:

```
$browse_sub[0][unactive] = "<li><a href=%URL>%NAME</a></li>";
```

Обратите внимание на слэши перед знаками $ и ", они обязательны, поскольку обработка этого массива должна осуществляться внутри функции, а не на этапе обработки макета дизайна.

В качестве наглядного примера шаблона можно привести шаблон вывода навигации Карты сайта. Одна функция строит полную иерархическую структуру сайта:

```
$browse_map['prefix'] = "<ul>";
```

```
$browse_map['unactive'] = "<li><a href=%URL>
%NAME</a>\".s_browse_sub(\$data[\$i][Subdivision_ID],\
$browse_template).\"</li>";
$browse_map['active'] = "<li><a href=%URL>
%NAME</a>\".s_browse_sub(\$data[\$i][Subdivision_ID],\
$browse_template).\"</li>";
$browse_map['suffix'] = "</ul>";
```

В данном примере каждый элемент шаблона вызывает функцию еще раз с одинаковым шаблоном, а в качестве номера раздела передается ID текущего выводимого раздела. Обратите внимание на слэши, они необходимы.

Почти все массивы из стандартных макетов являются необзятальными. Вы можете называть их как угодно. Обязательным массивом данных является $browse_msg. Если вы используете в компонентах функцию browse_messages, в макете дизайна обязательно должен быть данный шаблон вывода навигации, иначе функция ничего отображать не будет.

Пользовательские настройки в макетах

Помимо описанных выше возможностей макеты дизайна в NetCat поддерживают возможность настройки их отображения пользователем. Если один макет используется на разных сайтах и разделах, он может настраиваться в каждом конкретном случае. Для этого разработчик макета должен:

- описать настройки в специальном поле формы редактирования макета;
- внести изменения в макет в соответствие с этими настройками.

Настройки «видны» в хедере и футере макета в виде элементов массива $template_settings. Например, необходимо дать пользователю возможности:

- выбрать вариант выравнивания какого-то блока, например, логотипа;
- редактировать короткий текст, например, слоган;
- определять, будет ли выводиться какой-то блок, например, последние новости.

Для этого в поле «Настройки отображения макета в разделе» описываем эти настройки:

```
$settings_array = array(
    "ShowLogo" => array("type" => "select",
        "default_value" => "right",
        "caption" => "Отображать логотип",
```

```
            "validate_regexp" => "",
            "validate_error" => "",
            "values" => array("right" => "справа", "left" =>
    "слева"),
        ),
        "Slogan" => array("type" => "string",
            "default_value" => "Наша миссия отдыхать",
            "caption" => "Слоган",
            "validate_regexp" => "",
            "validate_error" => "",
            "size" => "20",
        ),
        "ShowNews" => array("type" => "checkbox",
            "default_value" => "",
            "caption" => "Выводить ли новости",
            "validate_regexp" => "",
            "validate_error" => "",
        ) );
```

В текст макета (футер или хедер) необходимо внести изменения:

```
…
<div align='".$template_settings[ShowLogo]."'><img src=…></div>
…
Наш девиз: ".$template_settings[Slogan]."
…
".opt(".$template_settings[ShowNews].", s_list_class(…))."
…
```

Эти настройки пользователь сможет изменять в форме редактирования настроек раздела или сайта, для которого определен данный макет.

Как видно из показанного выше примера, настройки отображения макета описываются в многомерном массиве $settings_array. Каждый элемент массива соответствует с одной стороны полю ввода (для пользователя), а с другой – элементу массива $template_settings[], доступному в макете.

Элементы массива:

- type: тип поля ввода. Варианты значений: string (строка), checkbox (независимый переключатель), select (выпадающий список) , textarea (текстовое поле), divider (разделитель для облегчения восприятия больших объемов настроек, подразумевается его использование в административной части сайта)

- default_value: значение по умолчанию (для checkbox следует всегда указывать false, т.к. при установке этого значения в true – нельзя будет установить значение false)
- caption: название поля для пользователя
- values: список значений для типа select, задается в виде массива array('ключ' => 'описание', ...)
- size: размер поля ввода. Для типа string – значение size html-элемента input; для типа textarea – значение rows html-элемента textarea
- validate_regexp: регулярное выражение для проверки корректности введённых данных. Применяется для типов string и textarea. Например значение /^\d+$/ будет означать ввод только цифровых символов. Подробнее о синтаксисе регулярных выражений можно ознакомиться в официальной документации к PHP.
- validate_error: сообщение об ошибках, выдаваемое при вводе данных не соответствующих регулярному выражению, описанному в validate_regexp

Примеры записи массива настроек с различными типами полей:

```
$settings_array = array(
     'Divider' => array('type' => 'divider',
          'caption' => 'Визуальные настройки'
     ),
     'Elements' => array('type' => 'string',
          'default_value' => '20',
          'caption' => 'Количество элементов на странице',
          'size' => '3',
          'validate_regexp' => '/^\d+$/',
          'validate_error' => 'Введите целое число'
     ),
     'Information' => array('type' => 'select',
          'default_value' => 'on',
          'caption' => 'Показать описание элемента',
          'values'   =>   array('on'   =>   'включить',   'off'   =>
'выключить')
     ),
     'Description' => array('type' => 'textarea',
          'default_value' => '',
          'caption' => 'Подробная информация',
          'size' => '5',
     ),
     'Comments' => array('type' => 'checkbox',
          'default_value' => '',
          'caption' => 'Доступны комментарии'
     )
);
```

Обязательные для указания элементы массива - type и caption, для типа select также values.

Чтобы использовать введеные настройки в макете, следует обращаться к ним через массив $template_settings. Для приведеного выше примера, обращение к настройкам можно описать следующим образом:

```
".( $template_settings['Information']=='on' ?
$template_settings['Description'] : "")."
...
".( $template_settings['Comments'] ? ... : "")."
```

Часть 6. Компоненты

Определение компонента дано в Руководстве пользователя. С точки зрения разработчика, компонент – это совокупность:

- данных, хранящихся в таблицах базы данных MySQL;
- описания структуры этих данных;
- правил (шаблонов) их отображения, в т.ч. выборки и фильтров, сортировки, вариантов отображения в зависимости от каких-либо факторов, разбиения списочных данных на страницы и пр.;
- шаблонов и правил добавления данных, их изменения, поиска по ним.

Настройка компонента и его полей используется для определения структуры данных компонента, внешнего вывода страниц, использующих этот компонент, внешнего вида форм добавления и редактирования записей данного компонента (объектов), определения действий после добавления, изменения и пр. объектов.

Если к какому-либо разделу прикреплен некоторый компонент, например, с пятью полями, то:

- при добавлении или изменении записей в разделе будет показана форма, состоящая из этих 5 полей, а также необязательные системные поля: приоритет (не играет роли, если в настройках компонента заполнено поле «Сортировать объекты по полю», либо параметр сортировки переназначен каким-либо иным способом), признак включенного объекта (если он установлен, объект показывается на сайте, и наоборот), а также ключевое слово (оно используется для адресации страницы с полным выводом информации об объекте). Стандартную форму добавления/изменения можно переопределить в соответствующем шаблоне действий для этого компонента;

- страницы будут показаны в том формате, который определяется данным компонентом: сначала будет отображен префикс, потом список объектов в формате, определенном в шаблоне вывода объекта, затем суффикс;

- если в шаблоне вывода объекта предусмотрена ссылка на страницу полного ее вывода (выражение вида подробнее или подробнее), по этой ссылке для каждой записи будет показана страница с данной записью в формате, определенном в шаблоне полного вывода объекта;

- после добавления объекты будут появляться в разделе сразу (если установлен режим публикации после добавления) или после включения их (если установлен режим публикации после проверки);

- если количество объектов компонента превышает число объектов на странице, указанное в настройках этого компонента, объекты будут отображаться порциями; для листинга по страницам используются переменные $nextLink и $prevLink;

- если не определен порядок сортировки объектов, по умолчанию они сортируются по внутреннему параметру «приоритет», дате добавления (последние добавленные объекты показываются первыми).

Создание и редактирование полей

Структура данных – набор полей для компонента – редактируется на странице списка полей в компоненте. Каждое поле имеет следующие характеристики:

- **Название поля.** Название поля в таблице MySQL и внутри системы. Допускаются латинские буквы, цифры, символ подчеркивания. Пример: «BookAuthor».
- **Описание.** Комментарий к полю. Пример: «Автор книги».
- **Тип поля.** Возможные варианты:
    - Строка – символьное поле;
    - Целое число;
    - Текстовой блок – мемо-поле (для ввода будет использован элемент формы <textarea>);
    - Список – список значений (для ввода будет использован выпадающий список <select>); возвращает значение указанной записи списка;
    - Логическая переменная – логическое поле (да/нет); при выводе записи возвращает 1 (да) или 0 (нет);
    - Файл – поле типа «файл» (для ввода будет использован элемент формы <input type=file>; возвращает URL файла;
    - Число с плавающей запятой;
    - Дата и время;
    - Связь с другим объектом;
    - Множественный список.
- **Формат.** Используется по-разному для полей различных типов:
    - Строка – **url, email, phone, password;**
    - Список – **латинское название списка**;

- Файл – указывается максимальный размер файла, его тип. Так же возможно указать тип файловой системы. Пример:«**25000:image/***»;
- Дата и время – «**event**», «**event_date**», «**event_time**».
- Связь с другим объектом – **«2:Name»**
- Множественный список – **латинское название списка**;

- Обязательно для заполнения. Устанавливает обязательность заполнения данного поля. Если поле не может быть пустым, объект данного компонента не будет добавлен/изменен в случае, если поле не заполнено.

- Возможен поиск по данному полю. Определяет, будет ли это поле участвовать в качестве аргумента для выборки по объектам этого компонента. Подробнее о поиске и выборке см. ниже.

- Приоритет. Определяет очередность вывода полей в формах добавления/изменения (если не определены шаблоны добавления и изменения).

- Значение по умолчанию. Будет записано данное значение, если поле не заполнено (в этом случае поле не должно быть обязательным для заполнения). Если тип поля - «Логическая переменная», то при любом значении в этом поле параметру переменной будет присваиваться значение 1, т.е. «да», независимо от обязательности заполнения данного поля.

- Тип доступа к полю. Поле может быть доступно для записи всем, а может быть предназначено только для администраторов (например, для реализации функционала «вопросы-ответы» – любой может заполнять поле «вопрос», а поле «ответ» – только администратор). Также поле может быть закрыто для всех. Этот вариант используется в некоторых модулях – в поле автоматически записывается некоторая информация – например, количество показов баннера.

Типы полей компонента

Каждое поле компонента должно иметь какой-либо тип данных, которые поле будет содержать. Возможные варианты:

- Строка – символьное поле, максимально может содержать 255 символов.
- Целое число.
- Текстовой блок – мемо-поле (для ввода будет использован элемент формы <textarea>), может содержать 64 кб текста.
- Список – список значений (для ввода будет использован выпадающий список <select>); возвращает значение указанной записи списка.

- Логическая переменная – логическое поле (да/нет); при выводе записи возвращает 1 (да) или 0 (нет). Если у поля параметр «обязательно для заполнения» выключен, то будет выводиться блок radiobutton'ов (не важно, да, нет). Если этот параметр включен, будет выводиться checkbox.
- Файл – поле типа «файл» (для ввода будет использован элемент формы <input type=file>, возвращает URL файла, его настоящее название и размер.
- Число с плавающей запятой;
- Дата и время - 6 полей для хранения даты и времени;
- Связь с другими объектами. В компонентах выдает идентификатор связанного объекта, который можно вывести при помощи функции listQuery. Например формат поля «Связь с другим объектом» имеет значение «22», это означает, что связанные объекты берутся из компонента с идентификатором 22. Тогда получить объект можно следующим образом:

```
".listQuery("SELECT * FROM Message22 WHERE
Message_ID=$f_имя_поля", '$data[Title] //
$data[Date]')."
```

где «имя_поля» - имя поля связь с другим объектом. Привязать объект можно при добавлении или редактировании сообщения в разделе, к которому прикреплён компонент с таким полем.

Некоторые типы имеют форматы для кастомизации:

- Строка – можно не использовать поле «Формат». Если в формате указаны значения «**url**» или «**email**», при добавлении/изменении значения в поле будет проверяться его соответствие формату URL и электронной почты соответственно; если указать значение «**password**», то при вводе данных в поле будут отображаться звездочки. Если в формате указан «**phone**», то значение будет проверяться как телефонный номер. Телефонный номер состоит из 7 цифр с необязательным кодом города (он может быть в скобках) и кодом страны (перед ним может стоять знак «+»). Группы цифр могут разделяться символом «-», пробелы и знаки табуляции игнорируются. Примеры правильного заполнения поля с форматом «**phone**»:
  - 1234567
  - 123-45-67
  - 8-123-45-67

- 8 (000) 123-45-67
- 8-000-123-45-67
- +7 000 123 45 67

- Целое, с плавающей запятой, логическое – формат не используется;
- Текстовый блок – в поле «Формат» необходимо указать высоту и ширину элемента <textarea>, который будет использоваться для добавления/изменения поля объекта через двоеточие (например, «**8:40**»);
- Список – в поле «Формат» необходимо указать **латинское название списка** (название таблицы), содержимое которого будет использоваться при выводе списка возможных значений;
- Файл – в поле должен быть указан максимальный размер файла в байтах. При необходимости может быть указан также и возможный тип файла (mime type) – к примеру, формат «**25000:image/***» означает, что размер файла не может превышать 25000 байт, при этом файл должен быть картинкой. Или «**25000**» означает, что размер файла не может превышать 25000 байт.
- Дата и время – в поле «Формат» можно указать «**event**», тогда при добавлении записи в это поле будет автоматически подставляться текущая дата и время при условии, что поле обязательно для заполнения. Можно указать «**event_date**», тогда в форме добавления/изменения будут показываться только поля даты (без времени), и при добавлении записи в это поле будет автоматически подставляться текущая дата при условии, что поле обязательно для заполнения. Можно указать «**event_time**», тогда в форме добавления/изменения будут показываться только поля времени (без даты) и при добавлении записи в это поле будет автоматически подставляться текущее время при условии, что поле обязательно для заполнения.
- Связь с другими объектами – в поле «Формат» следует указать номер компонента для привязки, если связь осуществляется с компонентами. В случае связи системных таблиц, указывается название системной таблицы (Subdivision, Sub_Class, User, Catalogue...). Через двоеточие указывается опциональный параметр для заголовка объекта в форме и списке объектов, если не указан, для компонентов будет "<тип объекта> #123". Примеры правильного оформления поля «Формат» для связи с другими объектами:

- **1** – *номер компонента*

- **2:Name** – *номер компонента и заголовка объекта*
- **2:"CONCAT(Message_ID, ': ', Announcement)"** – *номер заголовка объекта и расширение запроса*
- **Subdivision** – *имя системной таблицы*
- **User:CONCAT(FirstName, ' ', LastName)** – *имя системной таблицы и расширение запроса*

Расширение запроса используется для получения данных об объекте, например заголовка. Расширяемый запрос имеет вид:

```
SELECT <ID_Field> AS ItemID, CONCAT( [...] ) AS
ItemCaption FROM <Table> WHERE <Constraints>
```

Тип поля «множественный выбор»

Данный тип поля (компонента, раздела, сайта, пользователей) позволяет пользователю выбирать несколько элементов из списка.

Создание поля

В «типе поля» при добавлении\редактировании поля компонента (или поля из системной таблицы) укажите «Множественный выбор».
Поле «Формат» в этом случае обязательно для заполнения. Оно должно содержать как минимум имя таблицы. Так же в поле «Формат» можно задать вид (select или checkbox) элемента, который будет выводиться по умолчанию в формах добавления, изменения и поиска.

По умолчанию используется select с высотой равной трем.
Для изменения элемента укажите его через двоеточие после имени таблицы.
При использовании select можно также задать высоту этого элемента (количество одновременно отображаемых элементов) опять же через двоеточие после слова «select»
Примеры заполнения поля формат:

Формат	**Результат**
Region	Элемнет select, size - 3
ShopUnits:checkbox	Элемнет checkbox
Manufacturer:select	Элемнет select, size - 3
ShopCurrency:select:2	Элемнет select, size - 2

Форма добавления, изменения

По умолчанию, как было сказано выше, элемент (select или checkbox) берется из формата поля. Т.е. Если вы используете форму «по умолчанию», то формат вывода элемента
будет взят из формата поля.
В альтернативных формах можно использовать функцию

string nc_multilist_field($field_name, $style = "", $type = "", $classID = "", $caption = false, $selected = false, $disabled = false, $getData = false)

Она возвращает строку с HTML-кодом.
Подробно это функция описана в Приложении, здесь остановимся на том, что третьем параметром можно указать тип элемента: select или checkbox, так же для элемента select можно задать высоту. Этот параметр имеет вид аналогичный, как и у формата поля, за исключением того, что здесь не нужно указывать таблицу. По умолчанию так же используется select с высотой 3.
Пример использования:

nc_multilist_field('countrySelect', '', 'checkbox', $classID, 1)

nc_multilist_field('countrySelect', '', 'select:5', $classID, 1)

Отображение элементов

При использовании компонента с полем «Множественный выбор», в списке отображения объектов и в полном выводе объекта доступна переменная $f_ИмяПоля, которая на самом деле является массивом, содержащим выбранные элементы, так же доступен массив $f_ИмяПоля_id, содержащим id выбранных элементов.

Для примера рассмотрим ситуацию, когда у объекта выбраны элементы из списка «Город» (Region):

1.Москва, 59.Екатеринбург, 110.Магадан.

Поле имеет имя city.

Тогда в массиве $f_city первый элемент ($f_city[0]) это «Москва», второй элемент «Екатеринбург» и третий - «Магадан», а массив $f_city_id имеет следующий вид:

$f_city_id[0] = 1, $f_city_id[1] = 59, $f_city_id[2] = 110

Для вывода этих элементов есть смысл перевести массив в строку. Это можно сделать с помощью функции nc_array_to_string (см. Приложение).

Так же дле перевода массива в строку можно использвать функцию php join (подробнее см. http://ru2.php.net/join).

Все это справедливо и при использовании поля типа «Множественный выбор» в системных таблицах.

Поиск

По умолчанию, в форме поиска показывается элемент в соответствии с форматом этого поля.

Полю «Множественный выбор» соответствует два элемента массива srchPat[]

Первый элемент — это массив всех искомых элементов(либо строка с элементами, разделенными дефисом), второй — тип поиска.
Для этого типа поля существует три типа поиска:
1. Неполное совпадение (используется по умолчанию) — выберутся только те объекты, содержащие все искомые элементы, причем объекты так же могут содержать и другие элементы.
Элемент массива srchPat в этом случае равен 0
2. Полное совпадение — выберутся объекты содержащие только искомые элементы.
Элемент массива srchPat в этом случае равен 1
4. Частичное совпадение (хотя бы один) — выберутся объекты, содержащие хотя бы один их искомых элементов.
Элемент массива srchPat в этом случае равен 2

Для наглядности рассмотрим пример: пусть имеются четыре объекта: cites1, cites2, cites3, cites4 компонента, содержащего поле «Множественный выбор» формата Region (города). Для простоты, будем считать, что только по этому полю возможен поиск.

Содержание объектов
cites1 : 1.Москва, 59.Екатеринбург, 110.Магадан
cites2 : 1.Москва, 84.Керчь, 110.Магадан
cites3 : 1.Москва, 93. Ковров
cites4 : 216. Челябинск, 110.Магадан

При типе поиска «Неполное совпадение» и искомых элементов «Москва», «Магадан» выберутся объекты cites1 и cites2.
В этом случает srchPat[0][]=1&srchPat[0][]=110&srchPat[1]=0, что аналогично srchPat[0][0]=1&srchPat[0][1]=110&srchPat[1]=0 или srchPat[0]=1-110&srchPat[1]=0

При типе поиска «Полное совпадение» искомых элементов «Москва», «Магадан» ничего не найдется, а если искомые элементами будут «Москва», «Ковров», то найдется только объект cites3.
Здесь srchPat[0][]=1&srchPat[0][]=93&srchPat[1]=1

При типе поиска «Хотя бы один»и при искомом элементе «Москва» найдутся cites1, cites2, cites3; при искомых элементах «Москва», «Магадан» выберутся все объекты.

В форме поиска по умолчанию используется первый тип поиска, тип поиска в этом случае передается через скрытое поле:

```
<input type='hidden' name='srchPat[x]' value='0'>
```

Если вы используете свою форму поиска, то можете сами выбрать тип поиска и тоже передать его через скрытое поле, например:

```
<input type='hidden' name='srchPat[x]' value='2'>
```

Возможен вариант, когда пользователь сам может выбрать тип поиска, для этого в форму поиска вставьте:

```
<input type='radio' name='srchPat[x]' value='0'> Неполное совпадение
<input type='radio' name='srchPat[x]' value='1'> Полное совпадение
                <input type='radio' name='srchPat[x]' value='2'> Хотя бы один из выбранных
```

Создание и редактирование шаблонов вывода

Создание компонента подразумевает автоматическое создание новой таблицы в базе данных. Структура таблицы определяется набором полей для нее. Для добавления компонента (или его редактирования) необходимо заполнить соответствующую форму (см. «Руководство пользователя»).

Четыре макета (префикс, суффикс, макет вывода объекта в списке и полного вывода) представляют собой HTML-текст со вставками функций PHP, системных функций NetCat и специальных переменных. Так же, как и

содержимое макетов дизайна, эти макеты выполняются при помощи функции eval(): eval("echo $template;"). Поэтому к ним применимы все те же правила, что и для макетов дизайна: маскировка кавычек и обратного слеша, сцепление строк при помощи точки и пр. В Приложении 2 приведен список функций и переменных, которые можно использовать в макетах и компонентах.

Префикс и суффикс обычно используются в следующих случаях:

1. Если компонент подразумевает какой-либо HTML-текст перед списочной частью (или после нее), он указывается в макетах префикса и суффикса соответственно. Например, теги <table…> и </table> (а также, возможно, заголовочная часть страницы), если данные на страницах компонента должны представляться в табличном виде.

2. Если компонент содержит элементы навигации по компоненту раздела (если подразумевается, что список будет многостраничным). Например, необходимо реализовать навигацию в следующем виде:
Новости 11 - 20 из 35 *пред.* | *1* | ***2*** | *3* | *след.*

Для реализации этого примера необходимо внести в суффикс (или префикс, что встречается реже) примерно следующий текст:

```
Новости $begRow - $endRow из $totRows    
".opt($prevLink, "<a href=$prevLink>пред.</a>")."
".browse_messages($cc_env, 10)." ".opt($nextLink, "<a
href=$nextLink>след.</a>")
```

При этом настройки вывода блока навигации (поле «Шаблоны вывода навигации» в макете дизайна) должны быть следующими:

```
$browse_msg['prefix'] = "| ";
$browse_msg['suffix'] = " |";
$browse_msg['active'] = "<b>%PAGE</b>";
$browse_msg['unactive'] = "<a title='%FROM - %TO' href=%URL>
%PAGE</a>";
$browse_msg['divider'] = " | ";
```

В этом примере ссылка «пред.» или «след.» будут показаны только в том случае, если мы находимся не на первой (или последней) странице списка.

Все записи в списке будут показаны в соответствии с макетом вывода записи. Он также представляет собой HTML-текст со вставками переменных и

функций, полный список которых приводится в Приложении 2. Для каждой записи определен набор переменных, каждая из которых соответствует полю компонента. Если Name – название поля в компоненте, то для ее отображения в макете вывода записи нужно указывать переменную $f_Name, т.е. к названию поля прибавлять префикс «$f_».

К любым переменным (в т.ч. и соответствующим полям компонента) можно применять любые доступные функции, например, opt($var, $output) (вывод $output в том случае, если $var не пусто и не равно нулю), is_even($var) (проверка на четность), стандартные функции PHP.

К примеру, необходимо вывести анкету сотрудника. Сущность «Сотрудник» состоит из полей:

- ФИО (Name, Строка, обязательное для заполнения)
- Дата рождения (Date, Дата и Время, обязательное для заполнения)
- Фото (Photo, Файл)
- Характеристика (Description, Текстовый блок, обязательное для заполнения)
- Отдел (Depart, Список)

Сортировать записи нужно по фамилиям сотрудников. Для этого в поле «Сортировать по полю» установим значение «Name». Вывод записей должен происходить в следующем виде:

```
Фото Фото Фото Фото Иванов Иван Иванович
Фото Фото Фото Фото Родился 32.13.1999, «Отдел маркетинга»
Фото Фото Фото Фото
Фото Фото Фото Фото Не пьет, не курит, не судим, женат, имеет трех
Фото Фото Фото Фото детей и автомобиль ВАЗ.
```

Данный пример может быть реализован примерно так.

Префикс:

```
<table width=100% border=0>
```

Суффикс:

```
</table>
```

Макет вывода записи:

```
<tr>
```

```
<td valign=top><img width=100 height=80 alt='$f_Name'
src=".opt_case($f_Photo, $f_Photo, "/images/nophoto.gif")."></td>
<td valign=top><b>$f_Name</b><br>
<i>Родился $f_Date_day.$f_Date_month.$f_Date_year."opt($f_Depart,
", отдел &laquo;$f_Depart&raquo;")."</i>
<br><br>$_Description</td>
</tr>
```

В приведенном примере префикс открывает таблицу, суффикс ее закрывает, а каждый объект представляет собой строку из 2х ячеек. В первой ячейке показывается фотография сотрудника с альтернативным текстом, соответствующим имени сотрудника. Если фотографии нет, показывается другая картинка (например, пустая или с надписью «Нет фото»). Во второй ячейке выводится имя сотрудника, дата рождения, затем отдел (если он есть) и потом характеристика. Обратите внимание: т.к. отдел является необязательным параметром, он проверяется на «не пустоту». Эту проверку можно было бы не осуществлять, но в тогда будет выведен некорректный текст:

```
…32.13.1999, отдел «»…
```

Поэтому весь текст, относящийся к необязательному параметру, нужно выводить только в том случае, если параметр не пустой. Если же в данном примере поле «Характеристика» было бы необязательным, код бы не изменился, т.к. отсутствие значения между «

» и «</td>» не повлияло бы на вид страницы.

Для демонстрации следующего приема усложним задачу. Пусть сотрудники должны выводиться по 10 человек на странице. Для этого установим соответствующий атрибут компонента в 10. Если оставить шаблоны компонента такими же, будут выведены первые 10 сотрудников, а на следующие страницы попасть будет нельзя. Для исправления этого необходимо ввести в суффикс или префикс элементы навигации по списку записей.

В NetCat поддерживается навигация по списку двух типов, которые могут совмещаться. В первом типе на каждой странице содержатся ссылки на следующую и предыдущую страницу. Для реализации этого типа навигации в суффикс должен быть введен примерно следующий код:

```
<tr>
<td><a href=$prevLink>назад</a></td><td align=right><a
href=$nextLink>вперед</a></td>
</tr>
</table>
```

Вместо надписей «назад» и «вперед» можно использовать и другие надписи, а также картинки (например, изображение стрелок). В приведенном примере на первой странице списка слово «назад» будет показано без ссылки, поэтому целесообразно проверять параметр $prevLink:

```
…".opt($prevLink, "<a href=$prevLink>назад</a>")."…
```

Аналогично следует поступить и с параметром $nextLink. Также можно использовать переменные $begRow, $endRow, $totRows, которые обозначают соответственно номер первой записи на странице, номер последней записи на странице и общее число записей на всех страницах.

Второй способ навигации подразумевает вывод списка всех страниц. Для его реализации нужно в суффикс или префикс внести функцию:
".browse_messages($cc_env,$range)."
$range – количество «страниц», выводимых функцией. Вместо этой переменной обычно пишется число. Подразумевается, что из множества страниц одновременно будет показываться только список из $range страниц. Например, Ваш листинг состоит из 20 страниц. Если $range=10, то, находясь на первой странице, Вы будете видеть страницы с 1 по 10; находясь на 15-й странице, Вы будете видеть страницы 10-20.

Внешний вид этого блока навигации должен быть настроен:

```
$browse_msg['prefix'] = "| ";
$browse_msg['suffix'] = " |";
$browse_msg['active'] = "<b>%PAGE</b>";
$browse_msg['unactive'] = "<a href=%URL>%PAGE</a>";
$browse_msg['divider'] = " | ";
```

Следующий прием одновременно показывает пример присваивания переменной значения и реализации такой частой задачи, как чередование формата вывода записей. Пусть необходимо выводить по две записи в строке. Для реализации этого примера нам потребуется ввести некую переменную-счетчик, четность которой необходимо проверять. Ввести счетчик нужно в префиксе, добавив в него строчку:

```
".opt($f_Counter=0, "")."
```

В данном случае функция opt() ничего не выводит, только присваивает начальное значение переменной $f_Counter (название может быть другим). В макете же вывода записи нужно:

- проверять четность счетчика и выводить в зависимости от этого код;
- прибавлять единицу к счетчику.

Для этого содержимое макета должно быть примерно таким:

```
".is_even($f_Counter, "<tr>")."
<td valign=top><img width=100 height=80 alt='$f_Name'
src=".opt_case($f_Photo, $f_Photo,
"/i/images/nophoto.gif")."></td>
<td valign=top><b>$f_Name</b><br>
<i>Родился $f_Date_day.$f_Date_month.$f_Date_year."opt($f_Depart,
", отдел \"$f_Depart\"")."</i>
<br><br>$_Description</td>
".is_even(!$f_Counter, "</tr>")."
".opt($f_Counter++,"")."
```

В первой строке, в случае четности записи, выводится тег <tr>, в предпоследней, в случае нечетности, выводится тег </tr>, а в последней к параметру прибавляется единица. Примерно так же можно организовать, например, вывод строк с разным цветом фона через одну.

Для демонстрации примера применения макета полного вывода еще усложним задачу. У каждого сотрудника должна быть еще и более полная характеристика (FullDescr, Текстовый блок, обязательное для заполнения), которая будет выводиться на отдельной странице при нажатии на ссылку «подробнее». Для этого добавим в макет вывода записи ссылку «подробнее» (ее можно продублировать, например, на фотографии или имени сотрудника):

```
…$f_Description <a href=$fullLink>подробнее</a></td>…
```

Ссылка будет иметь вид /about/persons/person_117.html, где /about/persons/ – адрес текущего раздела, person – ключевое слово компонента раздела, а 117 – номер объекта. Если при добавлении сотрудника было введено ключевое слово объекта, оно будет использовано в адресе страницы, который будет выглядеть так: /about/persons/ivanov.html. Пока шаблон отображения одного объекта на отдельной странице пуст, страница, куда можно попасть после нажатия на ссылку, также будет пуста. Для ее заполнения внесем следующий текст в этот шаблон:

```
<table width=100% border=0>
<tr>
<td valign=top><img width=100 height=80 alt='$f_Name'
src=".opt_case($f_Photo, $f_Photo, "/images/nophoto.gif")."></td>
<td valign=top>
```

```
<i>Родился $f_Date_day.$f_Date_month.$f_Date_year."opt($f_Depart,
", отдел \"$f_Depart\"")."</i>
<br><br>$_FullDescr</td>
</tr>
</table>
```

Чтобы вывести имя сотрудника в название страницы, укажем шаблон заголовка данного компонента:

```
Информация о сотруднике $f_Name
```

Стоит отметить, что в шаблон отображения объекта в списке необходимо добавить переменную $f_AdminButtons (в произвольном месте). В обычном режиме она пуста, а в режиме администрирования содержит статусную информацию о записи, ссылки «изменить», «удалить» и «включить/выключить». Также в шаблон (обычно в его начало) необходимо добавить переменную $f_AdminCommon. В режиме администрирования она будет выводить блок с технической информацией о компоненте и ссылками на добавление объекта и удаление всех объектов. Настройка дизайна этих административных блоков будет описана ниже.

Ссылки для действий с компонентом и с объектом формируются автоматически и записываются в переменные:
$addLink — ссылка на добавление объекта;
$searchLink — ссылка на поиск;
$subscribeLink — ссылка на подписку на данный компонент в разделе;
$rssLink — ссылка на rss-просмотр этой страницы (если доступно);
$xmlLink — ссылка на xml-выгрузки содержимого страницы (если доступно).
Так же в списке объектов доступны:
$editLink — ссылка на редактирование объекта;
$deleteLink — ссылка на удаление объекта;
$dropLink — ссылка на удаление без подтверждения;
$checkedLink — ссылка на включение \ выключение объекта;
$fullRSSLink — ссылка на rss-просмотр объекта;

Использование PHP-кода в компонентах

При настройке шаблонов данных вы можете использовать как API системы NetCat, так и практически любые средства самого PHP. Например, ".opt($f_Var,"Поле заполнено")." и ".substr($f_Var,0,3).". В первом случае используется функция API системы NetCat, во втором – стандартная функция PHP substr().

Однако очень часто необходимо использовать не одну функцию, а целый кусок PHP-кода. Такая возможность имеется. Для вставки PHP-кода в префиксе, суффиксе и объекте в списке используется следующая конструкция:

```
";
PHP-код
$result.= "
```

Вместо «PHP-кода» вставляется нужный вам код.

Для вставки PHP-кода в полном выводе объекта (Объект на странице) используется следующая конструкция:

```
";
PHP-код
echo "
```

Вместо «PHP-кода» вставляется нужный вам код.

Поиск и выборка

В системе компонентов реализована встроенная система поиска и выборки. Поиск можно производить по любым полям. Так, по текстовым и строковым (а также файловым – т.к. в базе данных хранится URL файла) полям производится поиск подстроки (без морфологии), по числовым – по диапазону значений, по логическим – да/нет, по спискам (классификаторам) – по конкретному значению.

Прежде всего, выборку можно осуществлять только по тем полям компонента, у которых установлен атрибут «возможен поиск по данному полю». Каждому из таких полей соответствует элемент (или два элемента) массива srchPat[], значения которого нужно подавать на скрипт, чтобы произвести выборку. Для того чтобы лучше понять принцип работы механизмов выборки, приведем примерный алгоритм осуществления поиска:

1. выбираем все поля, у которых установлено свойство «Искать по этому полю»;
2. сортируем их по возрастанию приоритета;
3. с каждым из них ассоциируем элемент массива srchPat[] по порядку, начиная с нуля; если тип поля – Int или Float, ассоциируем два элемента массива;
4. проверяем каждый элемент массива srchPat[], поданный на скрипт; если элемент не пустой – осуществляем выборку.

В следующей таблице приведен пример набора полей (только те поля, которые доступны для поиска) и соответствия им элементов srchPat[].

Приоритет	**Поле/название поля**	**Тип**	**Элементы srchPat[]**	**Содержание параметров**
1	Имя / Name	Строка	srchPat[0]	Подается искомая подстрока. Поиск по полю не производится, если параметр пуст.
5	Возраст / Age	Целое число	srchPat[1], srchPat[2]	Подается начало (1) и конец (2) диапазона поиска. Если один из параметров пуст, соответствующей границы диапазона нет. Если оба пусты, поиск не производится.
6	Отдел / Depart	Список	srchPat[3]	Подается номер записи классификатора. Если значение содержит ноль или пустует, поиск не производится.
18	Пол мужской? / IsMale	Логическая переменная	srchPat[4]	Подается единица («да») или ноль («нет»). Если значение пустое, поиск не производится.

Чтобы вызвать системную форму поиска для данного компонента раздела, нужно открыть страницу /sub/search_keyword.html, где sub – адрес раздела, keyword – ключевое слово компонента раздела.

Несколько примеров запросов:

1. Поиск только тех сотрудников, в имени которых есть подстрока «Bill»
…/staff/?srchPat[0]=Bill

2. Поиск только совершеннолетних сотрудников
…/staff/?srchPat[1]=16

3. Поиск сотрудников из некоторого отдела (которому соответствует, например, запись номер 4 в соответствующем классификаторе), которые не старше 60 лет
…/staff/?srchPat[3]=4&srchPat[2]=60

4. Поиск мужчин в возрасте от 20 до 30 лет
…/staff/?srchPat[1]=20&srchPat[2]=30&srchPat[4]=1

5. Поиск женщин по имени Jane из 4 отдела в возрасте от 30 до 35 лет
…/staff/?
srchPat[0]=Jane&srchPat[1]=30&srchPat[2]=35&srchPat[3]=4&srchPat[4]=0

Механизмы поиска можно применять как вручную (добавляя параметры в поле ввода строки URL), так и автоматически. Приведем несколько примеров:

1. **На титульной странице сайта нужно показывать список всех клиентов, у которых есть веб-сайт:**
Предположим, что мы имеем компонент «Клиенты», среди полей которого есть поле «Адрес сайта» (необязательное), и раздел со списком клиентов. Установим для этого поля возможность поиска и внесем в нужное место макета титульной страницы вызов функции s_list_class():

```
".s_list_class(1,2,"&srchPat[0]=http")."
```

В этом примере 1 – номер раздела, 2 – номер компонента раздела. Также можно варьировать внешний вид списка клиентов в зависимости от того, выводится он в своем разделе или на титульной странице (об этом см. в следующей главе).

2. **В некотором разделе первая страница должна представлять собой форму поиска по объектам:**
Для того чтобы реализовать этот функционал, необходимо в настройках компонента раздела (по которому осуществляется поиск) значение «Действие по умолчанию» изменить на «Поиск». После этого, при заходе в данный раздел в режиме просмотра, будет показываться форма поиска по полям, поиск по которым возможен (устанавливается при создании поля). По умолчанию выводится форма, сгенерированная системой. При желании, можно сверстать собственную форму поиска по полям в разделе «Компоненты», колонка «Шаблоны действий» (Поиск), «Форма расширенного поиска». При создании собственной формы поиска обязательно следует указать значение «index» для скрытого (hidden) поля «action». Действие формы (action) следует установить в ". $current_sub[Hidden_URL].", если компонент, по которому производится поиск, является компонентом по умолчанию (включен, наименьший приоритет) и ".$current_sub[Hidden_URL].$current_cc[EnglishName].".html

– для остальных компонентов. При этом метод запроса формы (method) должен быть «GET».

3. **Несколько подразделов раздела должны представлять собой выборку данных из родительского раздела:**
Подразумевается, что мы рассматриваем пример, приведенный в начале главы. Допустим, в разделе «Сотрудники» нам нужен список всех сотрудников, а каждый подраздел раздела «Сотрудники» должен содержать список сотрудников какого-либо отдела. Для этого создадим нужное количество разделов, соответствующее количеству отделов с аналогичными разделами, и укажем внешний URL следующим образом: «/staff/?srchPat[3]=1» для отдела, который соответствует первой записи в классификаторе «Отделы», «/staff/?srchPat[3]=2» для второй и т.д.

Использование условий и параметров

В предыдущей главе приводился пример, когда компонент сам создает условие и форматирует вывод записей в зависимости от этого условия. Компонент может также реагировать на внешние условия или переменные. Например, на титульной странице нужно выводить 3 последних новости (записи из раздела «Новости» соответствующего компонента) в сокращенном виде. Допустим, в компоненте «Новости» два поля: анонс новости и полный ее текст, записи выводятся по 10 новостей на странице, а сам компонент имеет следующие шаблоны:

- суффикс:

```
<a href=$prevLink>назад</a> Новости $begRow-$endRow из
$totRows <a href=$nextLink>вперед</a>
```

- макет вывода записи:

```
<b>$f_Date_day.$f_Date_month.$f_Date_year.</b> $f_Anons<br>
$f_FullText $f_AdminButtons<br><br>
```

Чтобы вывести 3 последних новости на титульную страницу, внесем в ее футер или хедер вызов следующей функции (пусть раздел «Новости» имеет номер 1, а номер компонента раздела – 2):

```
".s_list_class(1, 2, "&isTitle=1&recNum=3)."
```

После этого на титульной странице будут выводиться три последних новости, но в полном виде и с листингом. Теперь нужно настроить шаблон под использование переменной $isTitle:

- суффикс:

```
".opt(!$isTitle, "<a href=$prevLink>назад</a> Новости $begRow-
$endRow из $totRows <a href=$nextLink>вперед</a>")."
```

- макет вывода объекта в списке:

```
".opt_case($isTitle, "<b>$f_Date_day.$f_Date_month.
$f_Date_year.</b> $f_Anons<br><br>", "<b>$f_Date_day.
$f_Date_month.$f_Date_year.</b> $f_Anons<br>$f_FullText
$f_AdminButtons<br><br>")."
```

Другой пример: объекты должны разделяться чертой (тег <hr>). Можно выводить <hr> в конце каждого объекта (в макете вывода объекта в списке), но в этом случае после последней записи также будет черта. Для решения этой задачи следует проверять в макете, не является ли этот объект последним на странице:

```
…".opt($f_RowNum!=($recNum-1), "<hr>")."
```

Системные настройки компонентов

Поле "Системные настройки" представляет собой PHP-консоль и работает при отображении списка объектов. При помощи системных настроек можно, в частности, модифицировать SQL-запрос, возвращающий список объектов.

Изначально основной SQL-запрос, возвращающий список объектов на страницу, выглядит приблизительно следующим образом:

```
SELECT список_полей FROM MessageXX WHERE условия_выборки
```

Условиями выборки, в частности, могут являться:

- выбрать объекты только в текущем разделе
- выбрать объекты только в текущем компоненте раздела

- выбрать объекты только текущего пользователя
- выбрать только включенные объекты
- выбрать только родительские объекты

Ниже приведен набор флагов, влияющих на эти условия:

- $ignore_catalogue – если 1, игнорирует выборку объектов по текущему сайту (по умолчанию - 0);
- $ignore_sub – если 1, игнорирует выборку объектов по текущему разделу (по умолчанию - 0);
- $ignore_cc – если 1, игнорирует выборку объектов по текущему компоненту раздела (по умолчанию - 0);
- $ingore_user – если 1, игнорирует выборку объектов по текущему пользователю (по умолчанию - 1);
- $ignore_check – если 1, игнорирует выборку только включенных объектов (по умолчанию - 0);
- $ignore_parent – если 1, игнорирует выборку только родительских объектов (по умолчанию — 0);
- $ignore_all – позволяет создать свой запрос, вместо системного. При установке значения 1 - основной запрос не будет составлен системой и его можно будет скомпоновать вручную, посредствам использования переменных $query_select, $query_from и подобных (по умолчанию — 0);
- $ignore_limit - позволяет игнорировать ограничение на выбор объектов. При установке значения 1 - будут выбраны все объекты (по умолчанию — 0);
- $query_limit - позволяет задать ограничение на выборку объектов. Например "20, 10", в запросе отобразится "... LIMIT 20, 10".Использовать данный флаг можно только при включенном $ignore_limit или $ignore_all.
- $distinct – если 1, добавляет к запросу DISTINCT, т.е. получается SELECT DISTINCT fields FROM;
- $distinctrow – если 1, добавляет к запросу DISTINCTROW, т.е. получается SELECT DISTINCTROW fields FROM.

Параметры $ignore_catalogue, $ignore_sub и $ignore_cc весьма полезны для организации «сквозной» структуры данных. Например, чтобы вывести список новостей из раздела /news/ в разделе /important-news/, следует присвоить $ignore_sub = true и $ignore_cc = true, это позволит отобразить новости из обоих разделов (при условии, что в них используется один и тот же компонент новостей). Если речь идёт только о компонентах одного раздела - $ignore_sub использовать не нужно; соответственно, если в системе несколько сайтов, $ignore_catalogue = true сотрёт границы привязки по сайту.

Для подобных манипуляций непосредственно с прямыми ссылками на объекты, например /news/**news**_5.html и /important-news/**news**_5.html, следует учесть, что ключевое слово компонентов в разделе должно быть одинаковым, как показано выше.

> *Важно: идентификатор раздела и компонента в разделе, будут взяты из строки адреса, т.е. /important-news/ или /important-news/news_5.html, значения полей Subdivision_ID или Sub_Class_ID в базе будут проигнорированы.*
>
> *Главное правило в подобных манипуляциях: в используемых разделах, компонент должен быть один и тот же.*

Кроме этого, есть возможность модифицировать основной SQL-запрос путем добавления в оператор SELECT собственного кода. Код устанавливается в соответствующие переменные и автоматически вставляется в запрос. С учетом этих переменных (отмечены жирным шрифтом), основной запрос выглядит следующим образом:

```
SELECT a.список_полей,$query_select
FROM MessageXX AS a,$query_from
$query_join
WHERE условия_выборки AND $query_where
GROUP BY $query_group
HAVING BY $query_having
ORDER BY $query_order
LIMIT $query_limit
```

$query_limit будет задействована только вкупе с $ignore_limit или $ignore_all. $query_order и $query_limit НЕ используются в полном выводе объекта.

Переменной $query_select соответствует переменная $result_vars, содержащая соответственно список переменных, в которые необходимо записать значение выбираемых полей. Например, если $query_select содержит "b.CarType,f.Name", то $result_vars может содержать "\$cartype,\$name".

По умолчанию значения всех вспомогательных переменных не установлены.

В случае если Вам нужно вывести объекты с «подобъектами» (пример – иерархический форум), необходимо установить значение переменной $ignore_parent в единицу. В противном случае, отображаются только родительские объекты.

Кроме того, "Системным настройкам" доступны ассоциативные массивы $current_catalogue, $current_sub, $current_cc, $current_user и все переменные, подаваемые на загружаемую страницу методом GET либо в функции s_list_class().

Шаблоны действий

Для каждого компонента данных предусмотрено 14 шаблонов действий.

Форма добавления

По умолчанию поля в форме добавления выводятся по очереди в порядке возрастания приоритета по одному полю на строку. Эту форму можно увидеть, нажав на ссылку «добавить» в режиме редактирования раздела. Внешний вид формы можно переопределить. Это применяется, в частности, в тех случаях, когда добавлять записи могут внешние пользователи: чтобы придать форме особенный вид или скрыть некоторые поля, чтобы пользователь не мог их добавить (например, если в компоненте «вопросы и ответы» есть поля «вопрос» и «ответ», целесообразно в форме добавления оставить только вопрос). Для создания формы добавления возьмите HTML-текст стандартной формы и внесите в него нужные изменения.

Переменная $warnText содержит сообщение об ошибке добавления объекта. Вы можете поместить эту переменную в любом месте HTML-кода альтернативной формы добавления. Например, в начале формы:

".opt($warnText,"Ошибка: $warnText

")."

Внимание: в случае если компонент содержит файлы, при написании альтернативной формы добавления объектов необходимо указать атрибут ENCTYPE у тега FORM со значением "multipart/form-data" (<FORM ENCTYPE=multipart/form-data ...>), а для работы с полями типа Файл использовать функцию nc_file_field(), описанную ниже в «Приложении 2».

В данном поле вставка PHP-кода возможна лишь при помощи кавычек ".substr(…).".

Пример формы добавления для компонента с 2-мя полями: тип Файл и Список.

```
".opt($warnText, "Ошибка: $warnText")."
```

```
<form name='adminForm' enctype='multipart/form-data'
method='post' action='/netcat/add.php'>
<input name='admin_mode' value='1' type='hidden'>
<input name='cc' value='$cc' type='hidden'>
<input name='sub' value='$sub' type='hidden'>
<input name='catalogue' value='$catalogue' type='hidden'>
<input name='posting' value='1' type='hidden'>
<table border='0' cellpadding='2' cellspacing='0'><tbody>
<tr><td>Приоритет объекта:</td><td><input name='f_Priority'
size='5' maxlength='5' value='' type='text'> <input id='chk'
name='f_Checked' value='1'  ".opt($f_Checked,
"checked='checked'")." type='checkbox'> <label
for='chk'>включить</label></td></tr>
<tr><td>Ключевое слово:</td><td><input name='f_Keyword'
size='20' maxlength='255' value='$f_Keyword'
type='text'></td></tr></tbody></table>
<hr size='1'>
Файл:<br>
".nc_file_field("MyFile")."
<br><br>
Список:<br>
".nc_list_select("Gallery", "MyList", $f_MyList)."
<br>
Звездочкой (*) отмечены поля, обязательные для заполнения.<hr
size='1'>
<div align=right><input value=Добавить
type=submit></div></form>
```

Условия добавления

По умолчанию при попытке добавить объект поля проверяются согласно их свойствам (формату, типу и т.д.). Например, в случае если в числовое поле произведена попытка ввести строку, система запишет сообщение об ошибке в переменную $warnText и повторно выведет форму добавления. Некоторые дополнительные условия, которые нельзя настроить в свойствах полей (например, обязательность одного из двух полей), можно настроить в данном шаблоне действия.

Данное поле может содержать код языка PHP. В случае если произведена попытка добавить объект, значение полей которого не удовлетворяет условиям добавления, запишите в переменную $warnText сообщение об ошибке и установите значение переменной $posting в ноль (0). Например, если обязательно нужно заполнить одно из двух полей формы:

```
if (!$f_field1 && !$f_field2) {
     $warnText = "Необходимо заполнить Field1 или Field2!";
```

```
        $posting = 0;
    }
```

Действие после добавления

По умолчанию после добавления объекта появляется сообщение об успешном добавлении объекта. Содержание этой страницы (и действия, которые происходят после добавления) можно переопределить, например, для «вопросов-ответов» вывести сообщение о том, что в скором времени вопрос будет обработан. Также можно вызывать различные действия: операции с базой, отправку писем и пр. Все поля, отправленные через форму добавления, доступны в «действии после добавления».

Вставка PHP-кода в это поле осуществляется следующим образом:

```
";
PHP-код
echo "
```

В действия после добавления объекта компонента, при наличии поля типа список, доступна переменная $f_имяПоля_name — содержащая имя выбранного элемента. $f_имяПоля, как и $f_имяПоля_id содержат id.
Обоснование добавления: например, требуется послать письмо, не очень удобно самому выполнять sql-запрос для определения имени элемента классификатора.
В случае с множественным выбором доступны массивы с теми же именами.

Форма изменения, условия изменения, действие после изменения

Назначение и функциональность этих форм аналогичны форме изменения и действию после добавления. Разница лишь в действиях – эти настройки относятся к изменению, записи.

Внимание: в случае если компонент содержит файлы, при написании альтернативной формы изменения объектов необходимо указать атрибут ENCTYPE у тега FORM со значением "multipart/form-data" (<FORM ENCTYPE=multipart/form-data ...>) , а для работы с полями типа Файл использовать функцию nc_file_field(), описанную ниже в Приложении 2.

В данном поле вставка PHP-кода возможна лишь при помощи кавычек ".substr(…).".

Пример формы изменения для компонента с 2-мя полями: тип Файл и Список.

```
".opt($warnText, "Ошибка: $warnText")."
<form name='adminForm' enctype='multipart/form-data'
method='post' action='/netcat/message.php '>
<input name='admin_mode' value='1' type='hidden'>
<input name='cc' value='$cc' type='hidden'>
<input name='sub' value='$sub' type='hidden'>
<input name='catalogue' value='$catalogue' type='hidden'>
<input name='message' value='$message' type='hidden'>
<input name='posting' value='1' type='hidden'>
<table border='0' cellpadding='2' cellspacing='0'><tbody>
<tr><td>Приоритет объекта:</td><td><input name='f_Priority'
size='5' maxlength='5' value='' type='text'> <input id='chk'
name='f_Checked' value='1'  ".opt($f_Checked,
"checked='checked'")." type='checkbox'> <label
for='chk'>включить</label></td></tr>
<tr><td>Ключевое слово:</td><td><input name='f_Keyword'
size='20' maxlength='255' value='$f_Keyword'
type='text'></td></tr></tbody></table>
<hr size='1'>
Файл:<br>
".nc_file_field("MyFile")."
<br><br>
Список:<br>
".nc_list_select("Gallery", "MyList", $f_MyList)."
<br>
Звездочкой (*) отмечены поля, обязательные для заполнения.<hr
size='1'>
<div align=right><input value='Сохранить изменения'
type=submit></div></form>
```

Форма удаления

Это форма выводится при удалении объекта (а точнее, подтверждение удаления объекта). Такую форму можно получить, пройдя по ссылкам вида:
xxx.ru/news/delete_news_5.html
xxx.ru/netcat/message.phpcatalogue=1&sub=2&cc=7&message=5&delete=1

Форма по умолчанию выводит ссылку на удаление и ссылку для возврата в раздел.

В форме пишется HTML-код с возможностью вставки PHP-кода (как и в, допустим, списке объектов).
Для разрыва кода нужно использовать переменную $result:

```
форма_удаления
```

";
PHP-код
$result .= "

Условие удаления

Данное поле содержит код языка PHP и выполняется перед удалением.
Чтобы отменить удаление, нужно переменной $posting присвоить значение 0.
Текст, выводимой в этом случае, содержится в переменной $warnText.
Доступны те же переменные, что и при полном выводе объекта.

Данное поле можно использовать не только для отмены удаления объекта, но и для выполнения других действий, которые должны произойти при удалении. Например, удаление других объектов, логически связанных с удаляемым.

При пакетном удалении объектов доступен числовой массив $message.
Индекс и значение – номер удаляемого объекта. Отменить удаление конкретного объекта в этом случае можно функцией unset($message[НомерОбъекта]);

Действия после включения и удаления

Данные действия аналогичны действиям после добавления или изменения объекта.

Форма поиска

HTML-код, записанный в это поле, будет доступен в префиксе или суффиксе списка объектов в переменной $searchForm (если форма не задана, то переменная будет содердать форму поиска «по умолчанию»). Механизм работы поиска описан в главе Поиск и выборка. Если вам сложно сориентироваться, вы можете скопировать системную форму поиска для данного компонента и вставить ее код в данное поле. Чтобы вызвать системную форму поиска для данного компонента раздела, нужно открыть страницу /sub/search_keyword.html , где sub – адрес раздела, keyword – ключевое слово компонента раздела.

Форма расширенного поиска

Устанавливая для компонента раздела «действие по умолчанию» поиск, при просмотре мы увидим стандартную форму поиска по этому компоненту. вы

можете написать свой HTML-код для этой формы в поле «Форма расширенного поиска».

Текст письма для подписчиков, условия подписки

Поля зарезервированы для использования в некоторых модулях. Если в вашей системе установлены эти модули, см. документацию к ним.

Шаблоны компонентов

Шаблоны компонентов предназначены для вывода данных основного компонента, используя при этом поля (префикс, объект в списке, суффикс и т.п.) из другой сущности, которая и называется «Шаблон компонента».

Шаблоны компонента не содержат собственных объектов в базе данных и полей, все действия добавления, редактирования и удаления происходят с объектами основного, родительского, компонента. Внутри шаблонов компонента используются $f_ переменные основного компонента.

Всё это позволяет организовать различные варианты вывода одних и тех же данных. Каждый шаблон привязан к определённому компоненту, а также импортируются и экспортируется вместе с ним.

Чтобы вывести данные, используя определённый шаблон компонента, следует в административной части системы зайти в настройки шаблонов компонента и выбрать нужный шаблон.

Для задания шаблона компонента в выводе функции nc_objects_list(), следует в третьем параметре данной функции передать переменную nc_ctpl=ZZ, где ZZ — идентификатор шаблона компонента, например: s_list_class(XX, YY, "nc_ctpl=ZZ").

Более подробно о шаблонах компонента будет рассказано в следующей части.

Экспорт-импорт компонентов

Компоненты можно переносить из одной системы NetCat в другую посредством функционала экспорта-импорта компонентов.

Экспорт компонентов производится в разделе «Компоненты», где в форме редактирования каждого компонента есть ссылка «Экспорт». Создается tpl-файл, который предлагается скачать на локальный компьютер.

Структура tpl-файла:

- 1 строка – версия системы, для которой предназначен компонент
- 2 строка - комментарии
- 3 строка – создание таблицы Message для компонента
- 4 и дальше строки – создание полей для компонента в таблице Field

В разделе «Импорт компонента» созданный файл можно закачать, т.е. установить в систему.

Информация, содержащаяся в разделах, с экспортируемым компонентом не передается.

Оформление блоков администрирования $f_AdminButtons и $f_AdminCommon

Стандартное оформление блоков администрирования заложено в ядре системы, однако его можно изменить. Для этого в «Базовых настройках» имеются 4 дополнительных поля: включить/выключить пользовательские блоки, поле для AdminButtons, поле для AdminCommon, альтернативные параметры.

По умолчанию поля содержат копию системного оформления для удобного редактирования. В полях имеются несколько фиксированных переменных PHP, которые вам нужно оставить – именно они заменяются на нужные значения при прохождении через eval(). Поле «Дополнительные параметры» поможет вам, если вы передаете какие-то параметры на всех ссылках, однако это не распространяется на ссылки подверждения удаления объекта. Начните строку со знака & – и все указанные вами параметры добавятся к ссылке подверждения «Удалить объект» и «Вернуться в раздел».

$f_AdminButtons использует следующие переменные:

- $f_AdminButtons_id – ID объекта;
- $f_AdminButtons_priority – приоритет объекта;
- $f_AdminButtons_user_add – ID добавившего объект пользователя;
- $f_AdminButtons_user_change – ID изменившего объект пользователя;
- $f_AdminButtons_change – ссылка для изменения объекта;
- $f_AdminButtons_delete – ссылка для удаления объекта;
- $f_AdminButtons_check – ссылка для включения/выключения объекта.

$f_AdminCommon использует следующие переменные:

- $f_AdminCommon_cc – ID компонента раздела;
- $f_AdminCommon_cc_name – название компонента раздела;
- $f_AdminCommon_add – ссылка добавления объекта;
- $f_AdminCommon_delete_all – ссылка «удалить все».

Не забывайте экранировать кавычки, если вы их используете в HTML.

Пользовательские настройки в компонентах

Как и макет дизайна, компонент может быть настроен пользователем. Для этого в форме редактирования компонента предусмотрено специальное поле ввода «Настройки отображения компонента раздела». Синтаксис и схема использования настроек компонента полностью идентична аналогичным настройкам макета за одним исключением: настройки хранятся в массиве $cc_settings[], а не $template_settings[]. Форму же настройки пользователь увидит в свойствах компонента раздела.

Использование BB-кодов

BB-коды нужны для безопасной вставки HTML содержимого в альтернативные формы добавления. На данный момент используются 2 функции для управления BB-кодами:

```
nc_bbcode_bar($winID, $formID, $textareaID, $help="", $codes="",
$prefix="", $suffix="")
#
# Функция вывода панельки с BB-кодами над текстовым полем
#
# $winID - идентификатор окна для JS кода
# $formID - идентификатор формы для JS кода
# $textareaID - идентификатор textarea для JS кода
# $help - выводить строку с помощью?
# $codes - какие коды выводить, по-умолчанию все
# $prefix - префикс вывода панельки с кодами
# $suffix - суффикс вывода панельки с кодами
```

и

```
nc_bbcode($text, $cut_link="", $cut_full="", $codes="")
#
```

```
# Функция обработки текста с BB-кодами, применяется при выводе
списка объектов или при полном выводе объекта.
# Но может использоваться в любом другом месте системы.
#
# $text - текст для обработки
# $cut_link - ссылка на объект в полном просмотре (просто
$fullLink или $fullDateLink), для изменения формата парсинга кода
CUT
# $cut_full - режим полного просмотра объекта, для изменения
формата парсинга кода CUT
# $codes - какие коды выводить, по умолчанию - все
```

nc_bbcode_bar()

Допустим, нам требуется прикрутить панельку BB-кодов к альтернативной форме добавления. Это будет выглядеть следующим образом:

```
<form id='addPost' name='adminForm' enctype='multipart/form-data'
method='post' action='/netcat/add.php'>
  ...
  ".nc_bbcode_bar('this', 'addPost', 'blogMessage', 1)."
  <textarea rows='10' name='f_Text' id='blogMessage'>".
$f_Text."</textarea>
  ...
</form>
```

Для ограничения используемых кодов следует указать массив $codes:

```
...
".nc_bbcode_bar('this', 'addPost', 'blogMessage', 1, array("B",
"I", "U", "S") )."
...
```

nc_bbcode()

Для парсинга текста с BB-кодами, при выводе списка объектов, пишем:

```
...
".nc_bbcode($f_Text)."
...
```

Если используется код CUT, следует указать ссылку на полный просмотр объекта и идентификатор полного просмотра (если не в списке), иначе данный код будет проигнорирован:

```
...
```

```
<!-- при выводе объекта в списке -->
".nc_bbcode($f_Text, $fullDateLink)."
...
<!-- при полном выводе объекта -->
".nc_bbcode($f_Text, "", 1)."
...
```

Для ограничения используемых кодов следует указать массив $codes:

```
...
".nc_bbcode($f_Text, "", "", array("B", "I", "U", "S") )."
...
```

Описание BB-кодов:

SIZE - размер шрифта, не более 2 знаков, указывается в единицах pt, требуется закрывающий [SIZE=99]...[/SIZE]

COLOR - цвет шрифта в формате HexRGB (FFAA00), требуется закрывающий [COLOR=FF9900]...[/COLOR]

SMILE - разрешить смайлики

B - жирный шрифт, без параметров, требуется закрывающий [B]...[/B]

I - наклонный шрифт, без параметров, требуется закрывающий [I]...[/I]

U - подчёркнутый текст, без параметров, требуется закрывающий , требуется закрывающий [U]...[/U]

S - зачёркнутый текст, без параметров, требуется закрывающий [S]...[/S]

LIST - элемент списка, без параметров, требуется закрывающий [LIST]...[/LIST]

QUOTE - цитата сообщения, требуется закрывающий [QUOTE]...[/QUOTE] или [QUOTE='имя_пользователя']...[/QUOTE]

CODE - блок моноширинного текста, без параметров, требуется закрывающий [CODE]... [/CODE]

IMG - вставка изображения, [IMG='адрес_изображения']

URL - вставка гиперссылки, требуется закрывающий [URL]адрес_url[/URL] или [URL='адрес_url']текст_ссылки[/URL]

CUT - сокращение текста при выводе объекта в списке, требуется закрывающий [CUT]... [/CUT] или [CUT='текст_ссылки']...[/CUT]

Массив допустимых кодов $codes, все возможные значения:

```
array("SIZE", "COLOR", "SMILE", "B", "I", "U", "S", "LIST",
"QUOTE", "CODE", "IMG", "URL", "CUT")
```

Последовательность и количество элементов учитывается, т.е. если значению массива $codes будет соответствовать

```
array("LIST", "B", "U")
```

то будут учитываться только эти коды, в указанной последовательности.

Стили CSS

Для корректного отображения панельки с BB-кодами, следует прописать следующие стили в стили CSS макета дизайна. Данные стили предустановлены в стандартных макетах NetCat.

```
/* BBcodes bar & in text BBcodes */
select.nc_bbcode_bar_size {margin-bottom:5px; width:100px;}
img.nc_bbcode_wicon {border:0; width:27px; height:20px;}
img.nc_bbcode_icon {border:0; width:20px; height:20px;}
div.nc_bbcode_error {padding:3px 0; color:#AA0000; font-
weight:bold}
input.nc_bbcode_helpbox {margin:0 0 3px; padding:2px 0;
width:100%; font-size:10px; font-family:Verdana,Arial;
background:none; border:0;}
div.nc_bbcode_colors {position:absolute; background:#FFFFFF;
padding:3px; border:solid 1px #AAAAAA;}
div.nc_bbcode_color_top {white-space:nowrap;}
div.nc_bbcode_color {padding-top:2px; white-space:nowrap;}
input.nc_bbcode_color {padding:0px; cursor:pointer; height:20px;
width:20px; border:0px;}
input.nc_bbcode_color_white {padding:0px; cursor:pointer;
height:20px; width:20px; border:1px solid #AAAAAA;}
div.nc_bbcode_smiles {position:absolute; background:#FFFFFF;
padding:3px; border:solid 1px #AAAAAA;}
div.nc_bbcode_smile_top {white-space:nowrap;}
div.nc_bbcode_smile {padding-top:2px; white-space:nowrap;}
input.nc_bbcode_smile {padding:0px; cursor:pointer; height:22px;
width:22px; border:0px;}
img.nc_bbcode_smile_in_text {margin:0 0 -3px 0;}
/* BBcodes in text */
div.nc_bbcode_quote_1_top {margin:0px 25px 0px 25px;}
div.nc_bbcode_quote_1 {padding:20px; border:1px solid #CCCCCC;
background:#FFFFFF;}
div.nc_bbcode_quote_2_top {margin:0px 25px 0px 25px;}
```

```
div.nc_bbcode_quote_2 {padding:20px; border:1px solid #CCCCCC;
background:#FFFFFF;}
div.nc_bbcode_code {margin:10px 25px 10px 25px;}
span.nc_bbcode_list_closed {margin-left:1em; text-indent:-.65em;
display:block;}
div.nc_bbcode_list {margin-left:1em; text-indent:-.65em;}
span.nc_bbcode_color {}
span.nc_bbcode_size {}
a.nc_bbcode_url_1 {}
a.nc_bbcode_url_2 {}
img.nc_bbcode_img {}
span.nc_bbcode_s {}
a.nc_bbcode_cut_link {}
```

Работа с файлами и файловая система

В NetCat Вы можете использовать три типа хранения файлов:

- Простой
- Стандартный
- Защищенный

Тип файловой системы задается в настройках поля.
Рассмотрим каждую систему в отдельности.

Простая

Файлы хранятся в директории /netcat_files/. Имя файла на диске состоит из идентификатора поля, символа подчеркивания, идентификатора сообщения и расширения файла.

Стандартная

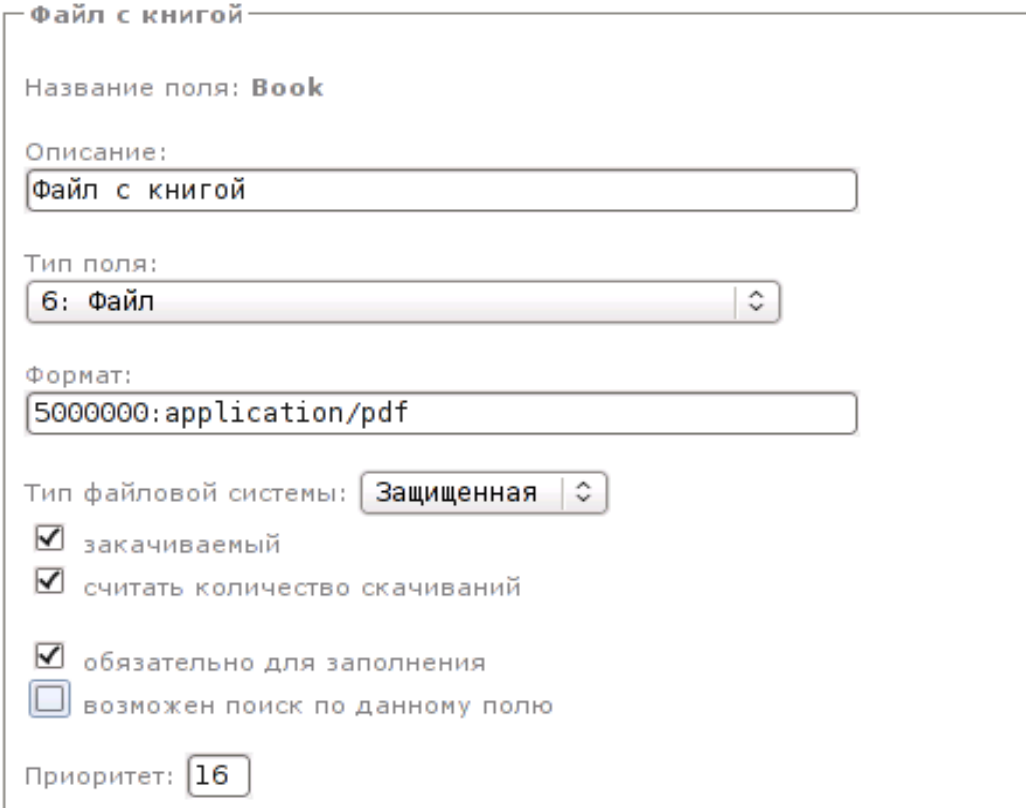
На диск записывается оригинальное имя файла (если оно содержит кириллицу — то символы переведутся в «транслит», если такой файл уже существует, то к новому добавится числовой индекс через подчеркивание). По умолчанию, файл записывается в директорию netcat_files/sub/cc/, где:
sub — номер раздела
cc — номер компонента в разделе

Вы можете поменять директорию, указав в условии добавления или изменение требуемое значение:
$f_ИмяПоля['folder'] = 'ИмяДиректории';

Имя директории должно содержать только латинские буквы. Для раз делений директорий между собой (для создания вложенной структуры) используется символ «/», например:
$f_Picture['folder'] = 'Image/foto';

Защищенная

Так же как и в случае с «стандартной» файловой системой, файлы записываются в директорию netcat_files/sub/cc/. На диск записывается захэшированное имя файла. Так же в таблице Filetable для каждого файла создается своя запись. Такой тип хранения файлов затрудняет получение файла подбором имени, но в этом случае создаётся нагрузка на базу данных. Для файлов этого типа можно задать атрибут «Закачиваемый». В этом случае при обращении к файлу браузер предложит его сохранить, а не показать. Так же для файлов данного типа можно считать количество обращений к ним, для этого надо задать атрибут «Считать количество скачиваний».



Пример поля типа «Файл»

Струкура таблица Filetable имеет следующий вид:
ID — индекс
Real_Name - настроящее имя файла
Virt_Name - виртуальное имя файла

File_Path - путь к файлу
File_Type - тип файла
File_Size - размер файла
Message_ID - номер объекта (раздела, пользователя, макета, сайта), к которому относится файл
Field_ID - номер поля файла
Content_Disposition - атрибут «Закачиваемый»
Download — количество скачиваний

Получение информации о файле в компоненте.
В списке объектов, при наличии поля типа «Файл», доступны переменные:
$f_ИмяПоля — ссылка на файл;
$f_ИмяПоля_url — прямая ссылка на файл (для ФС типа protected);
$f_ИмяПоля_name — оригинальное имя файла;
$f_ИмяПоля_size — размер файла;
$f_ИмяПоля_type — MIME-тип файла;
$f_ИмяПоля_download — количество запросов (скачиваний) файла (при ФС «Защищенная»).

В условие добавления:
$f_ИмяПоля — массив с ключами name, size, type

В действии после добавления:
$f_ИмяПоля — массив с ключами name, size, type
$f_ИмяПоля_name — оригинальное имя файла
$f_ИмяПоля_size — размер файла
$f_ИмяПоля_type — MIME-тип файла

Работа с файлами для сайтов, разделов, макетов дизайна и пользователей аналогична.

Часть 7. Шаблоны компонента

Описание

Шаблоны компонента позволяют выводить объекты компонента в разных видах.

Рассмотрим пример с каталогом товаров. вы хотите выводить новинки из каталога товара на титульную страницу, выводить в соседнем разделе прайс-лист товаров, создать rss-ленту с последними поступившими товарами… По существу, во всех случаях будут выводиться объекты одного компонента «Каталог товаров», но в разных форматах. Выводить таким образом объекты с помощью условий в рамках одного шаблона крайне затруднительно. Чтобы облегчить такую задачу разработчику, NetCat позволяет для каждого компонента создать неограниченное количество шаблонов вывода объектов — шаблонов компонента. В данном примере целесообразно было бы создать шаблоны «Новинки», «Прайс-лист», «RSS». В конце этого раздела мы более детально рассмотрим этот пример.
Следует отметить, что шаблоны компонента не имеют своих объектов, они лишь выводят объекты своего компонента, просто в другом формате.

Типы шаблонов компонента

По функциональному назначению можем разделить шаблоны на типы:

- Обычные
- Режим администрирования
- Режим редактирования
- RSS
- XML

Обычные шаблоны компонента используются для вывода объектов компонента в различных местах на сайте в нужном виде.

Шаблоны компонента для «режима администрирования» используются при просмотре раздела в системе администрирования.
К примеру, если у вас есть компонент «Новости», прикрепленный к разделу «Новости компании», а у него есть шаблон компонента для «режима администрирования», то при просмотре через систему администрирования объекты будут выводиться именно так, как задано в шаблоне компонента. Если

шаблона компонента для «режима администрирования» нет, то будет использоваться сам компонент «как есть».

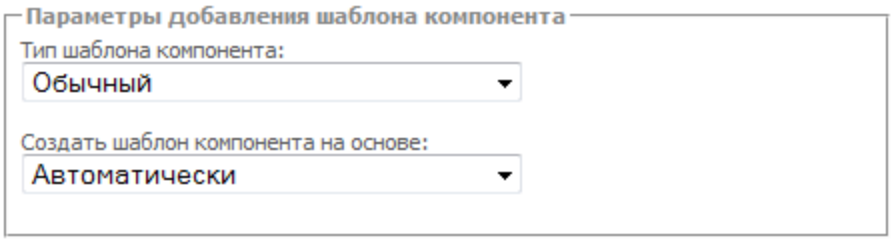
Шаблоны компонента для «режима редактирования» используются при работе через фронт-офис (чтобы попасть во фронт-офис, достаточно зайти по адресу http://адрес_сайта/netcat/). Работа во фронт-офисе так же называется «режим редактирования»)

Шаблон компонента «RSS» используется для организации новостных лент на сайте. Если такой шаблон есть, и в настройках компонента раздела включено использование rss, то по адресу:
http://ваш_сайт/раздел1/../разделN/КлючевоеСловоКомпонента.rss
будет выводиться новостная лента.

Шаблон компонента «XML» используется для организации xml-выгрузок объектов. Этот шаблон будет выводиться на странице вида
http://ваш_сайт/раздел1/../разделN/КлючевоеСловоКомпонента.xml

Создание и редактирование шаблонов компонента

Для создания шаблона компонента необходимо в системе администрирования перейти на нужный компонент и нажать «Добавить шаблон» (кнопка находится снизу слева). После этого вы увидите параметры создаваемого шаблона компонента.



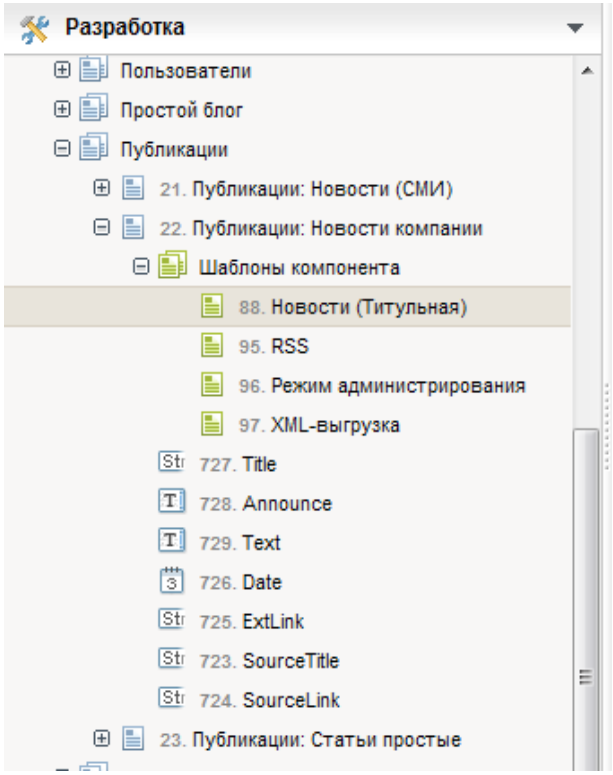
Параметры добавления шаблона компонента

В поле «Тип шаблона компонента» выберите нужный (все типы подробно рассмотрены выше: обычный, режим редактирования, режим администрирования, rss, xml).

Вы можете создать «пустой» шаблон компонента: как на основе уже существующего компонента, так и автоматически, системой. В последнем случае система NetCat сама сформирует шаблон компонента; например, при

создании шаблона типа «rss» автоматически появятся обязательные теги и предполагаемые поля.

Редактировать шаблоны компонента можно точно так же, как и сами компоненты. В дереве слева под каждым компонентом отображаются его шаблоны (если таковые имеются):



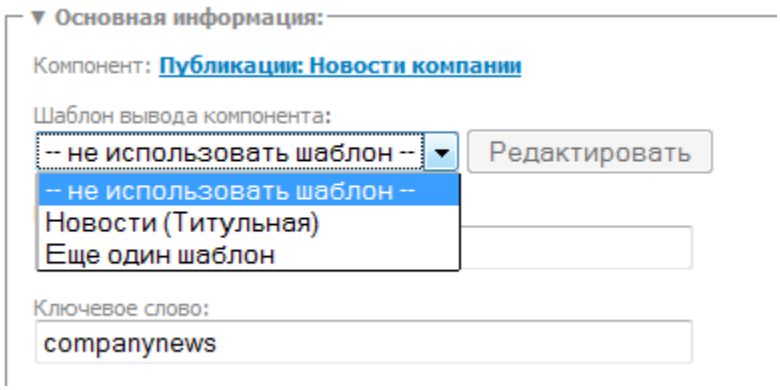
Компонент и его шаблоны

Использование шаблонов компонента

Шаблоны типа «RSS» и «XML» автоматически применяются к страницам, адрес которых имеет вид
http://ваш_сайт/раздел1/../разделN/КлючевоеСловоКомпонента.rss
и
http://ваш_сайт/раздел1/../разделN/КлючевоеСловоКомпонента.xml
соответственно
(по ссылке приведен адрес отображения списка объектов компонента, только оканчивается он не на html, как обычно, а на rss и xml).
При заходе на такую страницу макет дизайна не подгружается.
Если пользователь заходит на такую страницу, но соответствующего шаблона компонента нет, то он увидит ошибку «404. Страница не найдена».

Шаблоны типа «режим администрирования» и «режим редактирования» автоматически подгружаются в системе администрирования и во фронт-офисе соответственно. Если соответствующих шаблонов нет, то будет использован сам компонент «как есть».

При создании или редактировании компонента в разделе можно указать, какой шаблон компонента будет использоваться. Если шаблон не будет задан, то будет использоваться сам компонент.



Выбор шаблона компонента в настройках компонента в разделе

Если вам необходимо вывести объекты компонента в макете дизайна или из другого компонента с помощью функции nc_objects_list (аналог s_list_class) и вы хотите указать шаблон компонента для вывода, то в третьем параметре функции необходимо передать значение nc_ctpl равный номеру шаблона, например:

```
".nc_objects_list(12,30, "nc_ctpl=25&recNum=5")."
```

В этом случае будут отображены 5 объектов раздела с номером 12 и компонента в разделе с номером 30, причем для вывода объектов будет применяться шаблон компонента с номером 25.

Макропеременные, доступные в шаблонах

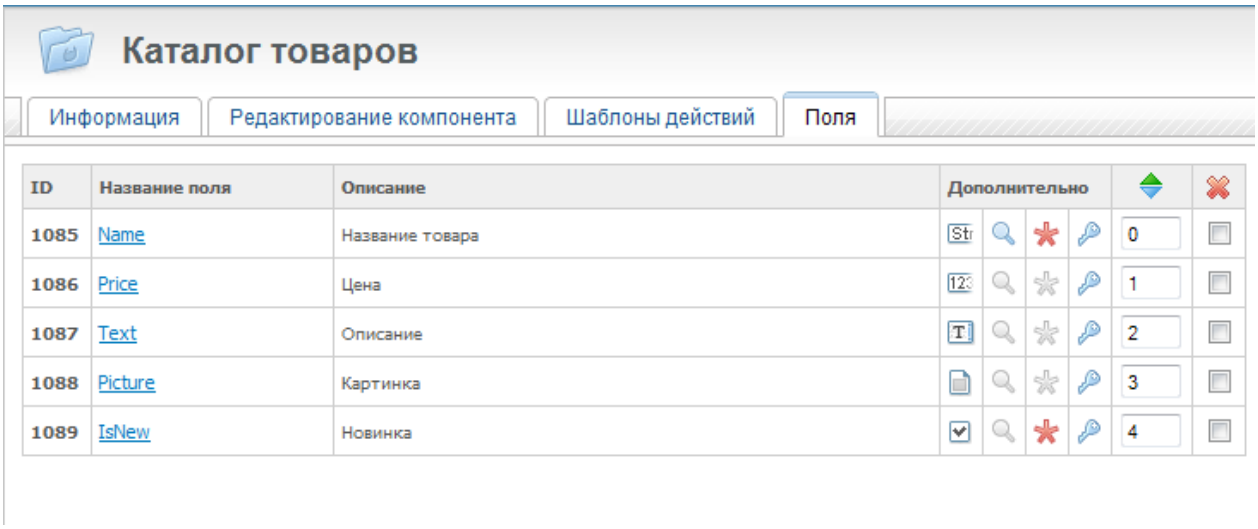
Иногда в шаблоне компонента необходимо получить знание какого-либо свойства самого компонента. Например, «Системные настройки» или «Альтернативная форма добавления». Для этого в шаблонах компонента доступны макропеременные. Приведем их список и значение:

%Prefix% - префикс списка объектов компонента;
%Record% - объект в списке компонента;
%Suffix% - суффикс списка объектов компонента;
%Full% - отображение объекта на отдельной странице;

%Settings% - системные настройки компонента;
%TitleTemplate% - заголовок страницы при отображении одного объекта;
%Order% - поле сортировки объектов;
%AddForm% - альтернативная форма добавления;
%AddCond% - условие добавления;
%AddAction% - действие после добавления;
%EditForm% альтернативная форма добавления;
%EditCond% - условие изменения;
%EditAction% - действие после изменения;
%DeleteForm% - альтернативная форма добавления;
%DeleteCond% - условие удаления;
%DeleteAction% - действие после удаления;
%CheckAction% - действие после включения/выключения;
%SearchForm% - форма поиска перед списком объектов;
%Search% - форма поиска на отдельной странице.

Пример использования шаблонов компонента

А теперь подробнее продолжим рассматривать пример, который мы привели в самом начале раздела. Итак, у нас есть компонент «Каталог товаров» с полями:
«Названия товара» - «Name», строка, обязательно для заполнения, возможен поиск по данному полю;
«Цена» - «Price», целое число;
«Описание» - «Text», текстовый блок;
«Картинка» - «Picture», файл;
«Новинка» - «IsNew», логическая переменная, обязательно для заполнения.
Чтобы создать такой компонент, необходимо зайти в «Разработка» - «Список компонентов», нажать «Добавить компонент», выбрать «Создать компонент с нуля» и нажать «Добавить компонент». После чего перейти на вкладку «Поля», добавить приведенные выше поля и вернуться на вкладку «Редактирование компонента».



Список полей компонента

Для простоты примера будем использовать максимально упрощенную «верстку»: выводить товары в две колонки, в списке объектов покажем «Название товара», «Картинку» (если есть), «Цену» (если есть), а ссылку на подробный просмотр поставим на названии товара.

Код компонента:

Префикс списка объектов:

```
$f_AdminCommon
<div>
```

Объект в списке:

```
$f_AdminButtons
<div style='float:left; width: 250px; padding: 20px;'>
  ".opt($f_Picture, "<img style='width: 150px; height:150px; float: left;
margin-right: 20px;' src='$f_Picture' alt='$f_Name' />" )."
  <a href='$fullLink'><b>$f_Name</b></a>
  <br/>
  ".opt( $f_Price, "<span>Цена: $f_Price</span>" )."
</div>

".opt(  $f_RowNum % 2, "<div style='clear:both;'></div>" )."
```

Суффикс списка объектов:

```
</div>
```

Отображение объекта на отдельной странице вы можете заполнить сами.

Конструкция вида:

```
".opt( $f_Price, "<span>Цена: $f_Price</span>" )."
```

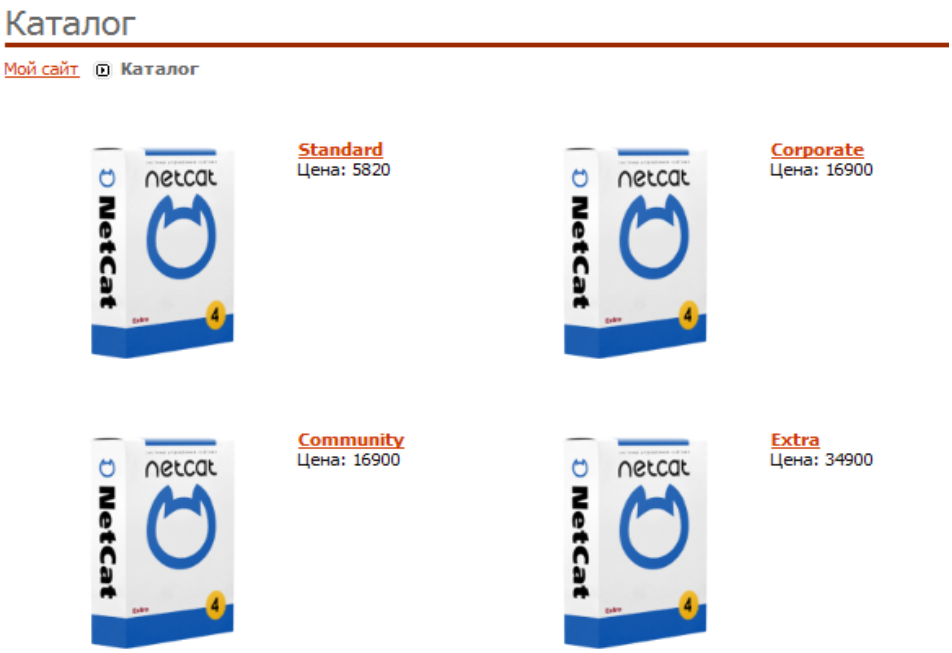
будет выводить
Цена: $f_Price
только в случае, если существует переменная $f_Price — то есть задана цена.

Конструкция:

```
".opt(  $f_RowNum % 2, "<div style='clear:both;'></div>" )."
```

выводит на каждом втором объекте

за счет этого достигается двухколоночность. $f_RowNum содержит номер текущего выводимого объекта по порядку начиная с 0.
Примечание: лучше не прописывать стили прямо в тэгах, ведь для этого есть файлы css, здесь же мы сделали так для упрощения.
Теперь создадим раздел с названием «Каталог» и ключевым словом «catalog» в корне сайта, прикрепим к нему наш компонент и добавим в него пару товаров (объектов).
После этого по ссылке http://ваш_сайт/catalog/ можно наблюдать каталог товаров.



Отображение раздела «Каталог»

Теперь создадим прайс-лист каталога в виде таблицы. Для этого добавим новый шаблон компонента «Каталог товаров», тип — обычный, создадим его «с нуля», то есть в поле «создать на основе» выберем «пустой», назовем его «Прайс-лист».

В префиксе шаблона компонента пропишем:

```
<table width='100%' border='1'>
<tr>
  <td width='70%'><b>Товар</b></td>
  <td width='30%'><b>Цена</b></td>
</tr>
```

Объект в списке:

```
<tr>
  <td><a href='$fullLink'>$f_Name</a></td>
  <td>$f_Price</td>
</tr>
```

Суффикс:

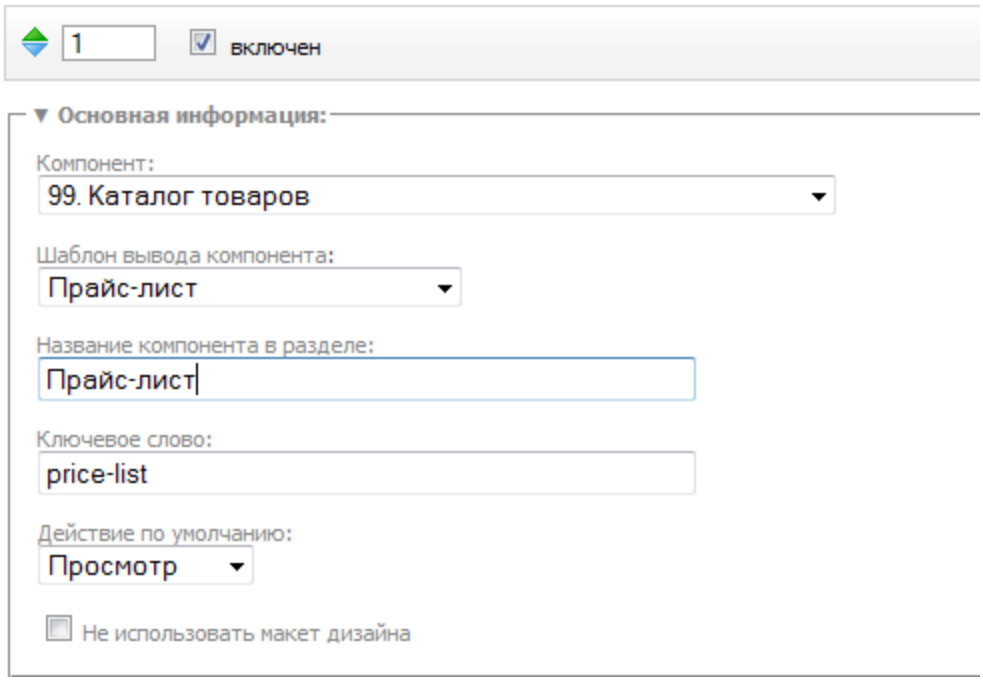
```
</table>
```

Системные настройки:

```
$ignore_cc = 1;
```

Шаблон компонента получился простой, запись в системных настройках нужна для того, чтобы вывести объекты «соседнего» компонента в разделе, где и находятся сами товары.

Создадим в разделе «Каталог» еще один компонент в разделе с ключевым словом «price-list», названием «Прайс-лист», в качестве компонента укажем «Каталог товаров», а так же укажем шаблон вывода компонента - только что созданный «Прайс-лист»:



Создание компонента в разделе

Теперь по адресу http://ваш_сайт/catalog/price-list.html будет доступен прайс-лист.

Теперь настроим вывод новинок на титульную страницу. Для этого создадим новый шаблон компонента с именем «Новинки на главную», тип - «обычный». В префиксе этого шаблона пропишем:

```
".opt( $totRows, "
<div>
 <b>Новинки</b>
")."
```

В объекте в списке:

```
<div>
 <a href='$fullLink'>$f_Name</a>
</div>
```

В суффиксе:

```
".opt( $totRows, "</div>")."
```

В системных настройках:

```
$query_where = "a.IsNew = 1";
```

$totRows содержит количество объектов, которые удовлетворяют заданному условию. Если новинок нет, то $totRows будет равным нулю и строки в префиксе

```
<div>
 <b>Новинки</b>
```

и в суффиксе

```
</div>
```

не выведутся.

Строка в системных настройках:

```
$query_where = "a.IsNew = 1";
```

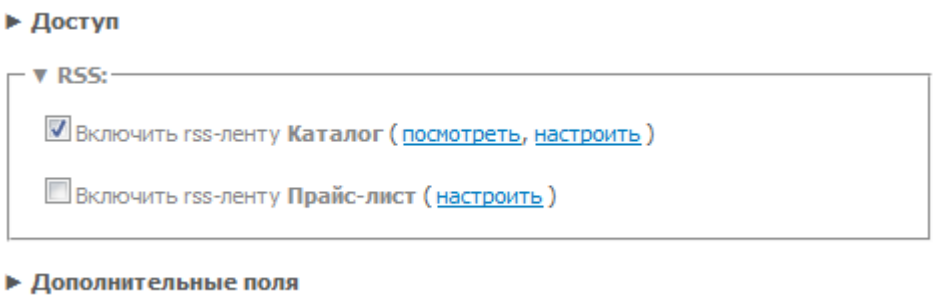
позволяет выбрать только новинки.

Теперь в макете дизайна титульной страницы нужно прописать:

```
".nc_objects_list(xxx, yyy, "nc_ctpl=zzz")."
```

где
xxx – номер раздела «Каталог»;
yyy – номер компонента в разделе «Каталог»;
zzz – номер шаблона компонента «Новинки на главную».

Давайте настроим также RSS-ленту товаров. Для этого создадим новый шаблон компонента, тип - «RSS», «создать на основе» - «автоматически». Система NetCat сама сформирует код компонента.
RSS надо включить, для этого в настройках раздела «Каталог» в области «RSS» нужно включить опцию «Включить rss-ленту Каталог»:



включение rss

Подправим префикс компонента «Каталог товаров», добавив туда ссылку на rss-ленту:

```
".opt($rssLink, "<a href='$rssLink'>RSS</a>")."
```

Теперь на странице http://ваш_сайт/catalog/catalog.rss будет доступна ваша новостная лента.

Для более удобного управления товарами через систему администрирования создадим новый шаблон компонента типа «режим администрирования» следующего содержания:

Префикс:

```
".( $searchForm ? "
<div id='nc_admin_filter'>
  <fieldset>
    <legend>Фильтр</legend>
    $searchForm
  </fieldset>
</div>
" : "" )."
```

Объект в списке:

```
<div>
  $f_AdminButtons
  ".opt($f_Picture, "<img style='width: 150px; height:150px; float: left;
margin-right: 10px;' src='$f_Picture' alt='$f_Name' />" )."
  <a href='$fullLinl'><b>$f_Name</b></a>
  <br/>
  ".opt( $f_Price, "<span>Цена: $f_Price</span>" )."
</div>
<div style='clear: both;'></div>
```

Данный шаблон будет отображать объекты в системе администрирования «построчно», добавив перед списком товаров фильтр для поиска.

Часть 8. Системный объект $nc_core

В системе до нынешнего момента присутствовал ряд сложностей, а именно:

- наблюдалась масса дублирующегося функционала с небольшими абберациями;
- система изначально была сделана в духе процедурного программирования, что не давало возможности осуществить грамотные взаимосвязи, из этого вытекала другая проблема — глобальные переменные;
- собственно, сами глобальные переменные, которые практически терялись своими корнями в большом количестве кода. Это было не совсем безопасно, замедляло отладку, разработку и расширение кода;

Основная цель введения данного объекта заключается в переводе системы на ООП (объектно-ориентированное программирование) и работе с новым API системы через этот объект. Сразу перевести систему на новые рельсы практически невозможно, поэтому не весь функционал данного объекта полностью соответствует принципам ООП.

Инстанцирование объекта происходит в файле **/netcat/system/index.php**:

```
// initialize superior system object
$nc_core = nc_Core::get_object();
// load default extensions
$nc_core->load_default_extensions();
```

Во второй строке вызывается статический метод get_object() класса nc_Core, который можно использовать в любом месте в системе, чтобы получить объект $nc_core. При повторном вызове метода будет возвращён уже существующий объект.

Подключение вышеописанного файла происходит в файле **/netcat/connect_io.php**, который загружается при любых операциях с системой, следовательно, и объект $nc_core всегда доступен в системе.

В четвёртой строке примера вызывается метод load_default_extensions(), который загружает основные расширения, необходимые для корректной работы с объектом.

Все классы (кроме **nc_Db** и **nc_System**) в папке **/netcat/system/** наследуют один абстрактный класс **nc_System**, который необходим для связи различных классов между собой и взят за основание.

*Заметка: все файлы классов названы по следующему шаблону — **nc_имякласса.class.php**. Имя класса в названии файла записано в нижнем регистре символов, в то время как название класса внутри файла написано с заглавной буквы — **nc_Имякласса**.*

Сам объект $nc_core, по сути, ничем не отличается от своих расширений, например, класса nc_Input. Но для удобства использования расширения не инстанцируются каждое по отдельности, а все объекты расширений попадают в объект $nc_core. Доступ к объекту расширения осуществляется следующим образом:

```
// get input extension
$nc_core->input;
```

Расширение nc_Essence предназначено для работы со стандартными сущностями системы — *Catalogue, Subdivision, Sub_Class, Class-Component, Template...* Так как каждая сущность отличается от остальных, то для удобства они поделены на классы расширения nc_Essence и располагаются в папке essences. Доступ к объекту сущностей осуществляется следующим образом:

```
// get input extension
$nc_core->сущность;
// for example
$nc_core->subdivision;
```

У каждого класса расширения есть определённые задачи и набор методов для их решения. Рассмотрим все, которые есть на данный момент.

Обращаем Ваше внимание, что данный функционал находится на этапе ранней разработки, поэтому может содержать спорные моменты.

Корневой абстрактный класс nc_System

Класс предназначен для связки объектов и для предоставления общего функционала, например, системы отладки.

public function errorMessage (Exception $e)

метод улавливания ошибки и её сохранения с данными объекта. Первый параметр — объект **Exception**.

public function debugMessage ($message, $file = "", $line = 0, $level = "info")

метод улавливания информации для отладки и её сохранения с данными объекта. Первый параметр — текст отладки, второй параметр — файл источник ошибки, третий параметр — строка в файле с ошибкой, четвёртый параметр — уровень отладочной информации (***info***, ***error***, ***ok***).

protected function debugInfo ()

метод формирует HTML код с отладочной информацией и ошибками.

Класс nc_Core *extends* nc_System

protected function __construct ()

метод конструктор, наследует родительский конструктор и устанавливает внутренние переменные.

public static function get_object()

статический метод для получения объекта. Если объект ещё не был инстанцирован, он инстанцируется.

public function set_variable ($name, $value)

метод доступа, позволяет задать системную переменную объекта. В параметрах передаётся имя переменной и её значение соответственно.

public function get_variable ($name)

метод доступа позволяет получить значение системной переменной объекта. В параметре передаётся имя переменной.

public function load ()

метод предназначен для загрузки расширений в систему. *На данном этапе используется только для конкретно заданных расширений, но в будущем будет доработан, дабы предоставить разработчику возможность загрузки своих расширений.*

public function get_settings ($item = "", $reset = false)

метод возвращает данные из таблицы настроек системы. При вызове метода без параметров будет возвращён ассоциативный массив, при указании в первом параметре имени интересующего поля из таблицы MySQL Settings — будет возвращено только одно конкретное значение. Второй параметр используется для сброса внутреннего статического кэша функции.

public function load_default_extensions ()

в методе загружаются системные расширения и классы сущностей (*Catalogue, Subdivision...*) посредством метода load(). *В будущем метод будет изменён.*

public function get_system_table_fields ($item = "")

пример метода дедубликации. Метод содержит в себе запрос на получение всех системных полей основных сущностей. Во всех случаях запрос одинаков. Возвращаемый результат содержит многомерный массив для всех системных сущностей. Если в первом параметре метода указать название определённой сущности, в результате будет возвращён одномерный массив с полями для указанной сущности.

public function load_env()

метод аналогичен устаревшей функции LoadEnv(). Пока данный метод не сильно оптимизирован и даже содержит в себе блок объявления глобальных переменных, что необходимо для сохранения совместимости.

Класс nc_Db *extends* ezSQL_mysql

Класс наследует другой класс — **ezSQL_mysql**.

public function set_connect ($dbname, $dbuser, $dbpassword = "", $dbhost = "localhost")

метод создаёт соединение с БД MySQL.

Абстрактный класс nc_Essence *extends* nc_System

Абстрактный класс сущностей.

public function get_current ($item = "")

метод возвращает данные текущей сущности. При вызове метода без параметров будет возвращён ассоциативный массив, при указании в первом параметре имени интересующего поля из таблицы MySQL **Имя_сущности** — будет возвращено только одно конкретное значение.

Текущая сущность является аналогом следующих: $current_catalogue, $current_sub и т.д.

public function set_current_by_id ($id, $reset = false)

метод задаёт данные текущей сущности по её идентификатору. Первый параметр — идентификатор, второй — флаг сброса локального статического кэша.

public function get_by_id ($id, $item = "", $reset = false)

метод получения данных сущности по её идентификатору. Первый параметр — идентификатор, второй параметр — конкретное поле их MySQL таблицы сущности, третий — флаг сброса локального статического кэша. Возвращаемый результат — массив или конкретное значение (при указании второго параметра).

Если сущность содержит наследуемые поля, данные наследуются.

public function delete_by_id ($id)

метод удаляет запись из таблицы сущности. Первый параметр — идентификатор сущности.

public function convert_system_vars ($env_array)

метод преобразует поля сущности в надлежащий вид. Это нужно для полей типа файл, дата, список и мультисписок. Первый параметр — массив данных сущности.

public function inherit_system_fields ($system_table_name, $parent_array, $child_array)

метод наследования полей от родительской сущности к дочерней. Первый параметр — имя системной сущности, второй параметр — массив данных родительской сущности, третий параметр — массив данных дочерней сущности. Результат — наследованный массив данных дочерней сущности.

Класс nc_Catalogue *extends* nc_Essence

Класс сущности **Catalogue**. Наследует все методы родительского класса **nc_Essence**.

public function get_by_host_name ($host, $current = false, $reset = false)

метод возвращает данные сущности **Catalogue** по адресу сайта. Первый параметр — адрес сайта, второй параметр — флаг установки данных в качестве текущих, третий параметр — сброс локального статического кэша.

Класс nc_Component *extends* nc_Essence

Класс сущности **Class** (он же **Component**). Наследует все методы родительского класса **nc_Essence**.

Класс nc_Message *extends* nc_Essence

Класс сущности **MessageXX**, где **XX** — идентификатор компонента. Наследует все методы родительского класса **nc_Essence**.

public function get_current ()

метод заглушка — всегда возвращает **false**, т.к. на данный момент не используется.

public function set_current_by_id ()

метод заглушка — всегда возвращает **false**, т.к. на данный момент не используется.

public function get_by_id ($class_id, $id, $item = "", $reset = false)

метод возвращает данные определённого сообщения. Первый параметр — идентификатор компонента, второй параметр — идентификатор сообщения, третий параметр — конкретное поле из MySQL таблицы сущности, четвёртый параметр — флаг сброса локального статического кэша. Результат — массив данных или значение определённого поля (если задан третий параметр).

public function delete_by_id ($class_id, $id)

метод удаляет запись из таблицы сущности. Первый параметр — идентификатор компонента, второй параметр — идентификатор сущности (сообщения).

Класс nc_Sub_Class *extends* nc_Essence

Класс сущности **Sub_Class**. Наследует все методы родительского класса **nc_Essence**.

public function get_by_subdivision_id ($id = 0, $reset = false)

метод возвращает данные сущности по идентификатору раздела. Первый параметр — идентификатор раздела (если он не задан, считается текущее значение раздела), второй параметр - флаг сброса локального статического кэша.

public function set_current_by_id ($id)

метод задаёт данные текущей сущности по её идентификатору. Первый параметр — идентификатор сущности. Этот метод отличается от родительского наследуемого метода, т.к. внутри устанавливает идентификатор $this->_current_id. Это необходимо, т.к. в разделе может быть несколько компонентов.

protected function inherit ($cc_env)

метод наследует данные из раздела, который, в свою очередь, наследует данные от сайта. Первый параметр — массив данных сущности.

public function validate_english_name ($str)

метод проверяет соответствие ключевого слова компонента в разделе системным требованиям. Первый параметр — ключевое слово для проверки. Результат — булевое значение результатов проверки.

Класс nc_Subdivision *extends* nc_Essence

Класс сущности **Subdivision**. Наследует все методы родительского класса **nc_Essence**.

public function set_current_by_uri ($reset = false)

метод устанавливает текущие данные сущности по адресу из системного массива $parsed_url. Для этого используется метод $nc_core->url->get_parsed_url('path'). Первый параметр - флаг сброса локального статического кэша.

public function validate_hidden_url ($url)

метод проверяет соответствие поля Hidden_URL раздела системным требованиям. Первый параметр — значение для проверки. Результат — булевое значение результатов проверки.

protected function inherit ($sub_env)

метод наследует данные от сайта. Первый параметр — массив данных сущности. Внутри метода устанавливаются системный массив $this->_parent_tree[***идентификатор***] и значение $this->_level_count[***идентификатор***] для данной сущности. Первый параметр — массив данных сущности.

public function get_parent_tree ($sub)

метод возвращает системный массив _parent_tree для данного раздела. Первый параметр — идентификатор сущности.

public function get_level_count ($sub)

метод возвращает значение _level_count для данного раздела. Первый параметр — идентификатор сущности.

Класс nc_Template *extends* nc_Essence

Класс сущности **Template**. Наследует все методы родительского класса **nc_Essence**.

public function convert_subvariables ($template_env)

метод заменяет субпеременные полей (***%имя_переменной***) на их значения. Первый параметр — массив данных сущности.
protected function inherit ($template_env)

метод наследует данные от родительских сущностей (макетов дизайна). Первый параметр — массив данных сущности.

Класс nc_User *extends* nc_Essence

Класс сущности **User**. Наследует все методы родительского класса **nc_Essence**.

Класс nc_Event *extends* nc_System

Класс событий.

public function bind (&$object, $events_remap_arr = false)

метод прикрепления объекта к событию. Первый параметр — следящий объект, второй — массив переназначения стандартных методов событий.

При возникновении события данные события передаются одноимённому методу прикреплённого объекта. Например, при событии **updateCatalogue**, будет вызван метод: $object->updateCatalogue(...). Если во время прикрепления события был назначен массив переназначения, то вызовется метод из массива.

Прикрепление с массивом переназначения:

```
// remap array
$nc_core->bind( $object, array("updateCatalogue" => "myMethod") );
```

public function execute ($event, $arg1, $arg2, ...)

метод создания события. Первый параметр — имя события, последующие параметры — данные, передаваемые в функцию слежения события.

Более подробное описание событий и передаваемых параметров находится в приложении.

Класс nc_Gzip *extends* nc_System

Класс поддержки сжатия gzip.

public function check ()

метод проверяет возможность использования сжатия браузером. Возвращает имя метода сжатия или false.

Класс nc_Input *extends* nc_System

Класс обработки входящих в систему данных.

public function clear_system_vars ($result)

метод удаляет из массива переданного в первом параметре все системные переменные. Первый параметр — массив входных данных.

public function prepare_extract ()

массив подготавливает данные из REQUEST массивов, очищает их и заносит в объект $nc_core. Результат — один массив обработанных данных. *В будущем этот метод можно использовать для более эврестической фильтрации входных данных.*

public function recursive_add_slashes ($input)

метод добавляет слэши к переданным ему данным. Первый параметр — переменная или массив. Результат — обработанный первый параметр.

public function fetch_get ($item = "")

метод доступа к обработанному GET массиву. Если не задан первый параметр — возвращается весь массив, если первый параметр задан и в нём содержится имя конкретного элемента массива — возвращается только это значение.

public function fetch_post ($item = "")

метод доступа к обработанному POST массиву. Если не задан первый параметр — возвращается весь массив, если первый параметр задан и в нём содержится имя конкретного элемента массива — возвращается только это значение.

public function fetch_cookie ($item = "")

метод доступа к обработанному COOKIE массиву. Если не задан первый параметр — возвращается весь массив, если первый параметр задан и в нём содержится имя конкретного элемента массива — возвращается только это значение.

public function fetch_session ($item = "")

метод доступа к обработанному SESSION массиву. Если не задан первый параметр — возвращается весь массив, если первый параметр задан и в нём содержится имя конкретного элемента массива — возвращается только это значение.

public function fetch_files ($item = "")

метод доступа к обработанному FILES массиву. Если не задан первый параметр — возвращается весь массив, если первый параметр задан и в нём содержится имя конкретного элемента массива — возвращается только это значение.

public function fetch_get_post ($item = "")

метод доступа к обработанному GET массиву и POST массиву. Данные из GET массива имеют приоритет при схождении перед данными из POST массива. Если не задан первый параметр — возвращается весь массив, если первый параметр задан и в нём содержится имя конкретного элемента массива — возвращается только это значение.

Класс nc_Lang *extends* nc_System

Класс поддержки мультиязычности.

public function full_from_acronym ($lang)

метод возвращает полное название языка по его акрониму. Первый параметр — акроним языка.

public function acronym_from_full ($lang)

метод возвращает акроним языка по его полному названию. Первый параметр — полное название языка.

Класс nc_Modules *extends* nc_System

Класс поддержки модулей системы.

public function get_data ()

метод возвращает ассоциативный массив данных для всех модулей.

public function get_by_keyword ($keyword, $installed = true)

метод возвращает ассоциативный массив данных для конкретного модуля. Первый параметр — ключевое слово модуля, второй параметр — флаг активности модуля, поле **Installed**.

public function load_env ($language = "", $only_inside_admin = false, $reload = false)

метод по сути повторяет устаревшую функцию LoadModuleEnv() загрузки модулей в систему.

public function get_module_vars ()

метод возвращает многомерный массив параметров для всех модулей.

public function get_vars ($module, $item = "")

метод возвращает параметры для конкретного модуля. Первый параметр — ключевое слово модуля, второй параметр — название конкретного параметра

модуля. Если второй параметр не указан — будет возвращён массив параметров, если второй параметр указан — будет возвращено конкретное значение.

Класс nc_Url *extends* nc_System

Класс работы с URL.

public function build_url ($query_arr)

метод возвращает строку параметров разделёнными амперсандами. Певый параметр — массив параметров.

public function parse_url ()

метод обрабатывает системный параметр REQUEST_URI и задаёт системный массив $parsed_url. Формат массива остался прежним, для сохранения совместимости.

public function get_parsed_url ($item = "")

метод доступа к многомерному системному массиву $parsed_url. Если первый параметр не указан — будет возвращён массив, если первый параметр указан — будет возвращено конкретное значение.

public function set_parsed_url_item ($item, $value)

метод доступа заменяет значения в системном массиве $parsed_url. Первый параметр — имя элемента массива, второй параметр — значение элемента массива.

public function get_uri_date ($timestamp = false)

метод возвращает дату из строки адреса, в соответствии с принципом работы системы. Если первый параметр не указан — будет возвращено строковое значение (***YYYY-MM-DD***), иначе — число в формате TIMESTAMP.

public function source_url ()

метод формирует полную ссылку на основе системного массива $parsed_url и возвращает её как результат.

Класс nc_Utf8 *extends* nc_System

Класс поддержки UTF8.

public function win2utf ($str)

метод конвертирует строку из кодировки cp1251 в кодировку UTF-8. Первый параметр — строка в кодировке cp1251. Результат — строка в кодировке UTF-8.

public function utf2win ($str)

метод конвертирует строку из кодировки UTF-8 в кодировку cp1251. Первый параметр — строка в кодировке UTF-8. Результат — строка в кодировке cp1251.

public function array_utf2win($input)

метод конвертирует массив данных из кодировки cp1251 в кодировку UTF-8. Первый параметр — массив в кодировке cp1251. Результат — массив в кодировке UTF-8. Если первый параметр — строка, будет возвращена обработанная строка.

Часть 9. Дополнительные инструменты

Мультиязычность

Система мультиязычности позволяет с минимальными усилиями перевести back-office системы (интерфейс администрирования) на требуемый язык. По умолчанию система NetCat содержит русский и английский языки.

Для подключения нового языка необходимо в папку /netcat/admin/lang/ положить файл вида **язык.php** (например, esperanto.php). Лучше скопировать уже существующий русскоязычный файл, переименовать его и перевести в нем все константы. Система подключит файл автоматически. В результате в интерфейсе администрирования в верхней части страницы появится переключатель на новый язык.

В блоке MAIN языкового файла содержится общая информация для работы с языком:

- MAIN_DIR - направление языка - LTR - Left-To-Right, RTL - Right-To-Left
- MAIN_LANG - 2-х символьный код языка по ISO 639-1
- MAIN_COUNTRY - 2-х символьный код страны
- MAIN_NAME - название языка на английском
- MAIN_NAMELOC - название языка в натуральном виде
- MAIN_ENCODING - кодировка данного файла и интерфейса администрирования
- MAIN_EMAIL_ENCODING - кодировка писем, которые будут оправляться из системы

Стоит учесть, что при добавлении нового языка потребуется добавить соответствующий язык и во все установленные модули. Файл вида **язык.lang.php** (находится в папке модуля) содержит все тестовые константы, используемые в модуле для обеспечения мультиязычности. Если Ваш сайт, скажем, имеет еще и немецкий интерфейс, Вам необходимо создать файл ger.lang.php (в папке / netcat/admin/lang/ должен присутствовать файл ger.php).

Пример содержимого файла язык.lang.php:

```
define("NETCAT_MODULE_AUTH", "Интерфейс пользователя");
define("NETCAT_MODULE_AUTH_DESCRIPTION", "Интерфейс пользователя в системе ввода-вывода. Возможность регистрации внешней группы
```

пользователей, изменение собственной анкеты, пароля, восстановление пароля. Данный модуль может интегрироваться с другими модулями системы.")

Управление задачами

Управление задачами позволяет автоматически запускать нужные скрипты в необходимое время.

Запускать можно локальные скрипты, либо скрипты, находящиеся на любом другом хостинге.

Настройка:

Для настройки данной функции необходимо отредактировать файл: netcat/admin/crontab.php . Вам нужно установить верные значения:

- $DOCUMENT_ROOT - физический путь до папки, содержащей папку netcat (например, /var/httpd/example/www)
- $HTTP_HOST – домен, на котором работает сайт (без http://, например, example.net)

Затем через панель управления хостингом (или иным способом, в зависимости от вашего провайдера), нужно прописать в crontab-файле файл netcat/admin/crontab.php на исполнение каждую минуту (если в списке задач есть ежеминутные, иначе периодичность можно подобрать иную).

Описание полей:

- Минуты - запускать каждые n минут
- Часы - запускать каждые n часов
- Дни - запускать каждые n дней **
- Последний запуск - время и дата, когда в последний раз запускался скрипт
- Ссылка на скрипт - относительная или полная ссылка на скрипт, который необходимо выполнить (http://example.net/index.php или /netcat/modules/optimize/cron.php)

* - ваш тарифный план должен поддерживать выполнение cron

** - если все три поля имеют значение 0, скрипт выполняться не будет вообще, минимальный интервал 1 минута.

Командная строка SQL

Функционал позволяет работать с БД при помощи SQL-запросов через веб-интерфейс. Вы можете создавать/удалять/изменять таблицы, удалять/изменять/добавлять поля, управлять ключами, выполнять любые SQL операции. Используйте этот функционал очень осторожно, т.к. фактически вы получаете полный доступ ко всей базе и неосторожным действием можете удалить или изменить любую информацию.

Визуальный HTML-редактор (WYSIWYG)

В системе предусмотрено 2 HTML-редактора: стандартный и FCKeditor.

Стандартный редактор позволяется работать в Internet Explorer (с поддержкой ActiveX) версии 5.5 и выше.

Редактор FCKeditor совместим с большинством популярных броузеров, и имеет больше фунций, нежели стандартный редактор. К тому же он может быть встроен непосредственно в страницу редактирования информации вместо вызова его в отдельном окне для каждого поля. FCKeditor является редактором по умолчанию.

Визуальный HTML-редактор присутствует на странице добавления/изменения записи для каждого компонента, у которого есть поля типа «Текст» и для которого разрешено использование HTML-тегов.

Выбрать необходимый вам редактор можно в разделе «Инструменты» - «Базовые настройки» системы администрирования. На странице присутствуют 2 параметра, один из которых отвечает за выбор редактора, а второй – за вариант его интеграции (доступен только для редактора FCKeditor).

В случае если редактор встроен в поле для редактирования (только для FCKeditor), интерфейс редактора будет непосредственно в каждом поле типа «Текст» прямо на странице редактирования объекта (только в том случае, если для компонента раздела разрешены теги).

Если Вы остановились на редакторе FCKeditor, то, помимо стандартных возможностей, он позволяет создавать собственный набор стилей для редактирования. На странице появляется поле "Набор стилей для редактора", в

котором можно создавать собственные стили. Информацию по структуре и синтаксису можно узнать на странице http://wiki.fckeditor.net/Developer%27s_Guide/Configuration/Styles.

Отслеживание ошибок

В процессе разработки проекта очень важно видеть те ошибки, которые возникают в ходе работы скриптов. В NetCat есть несколько механизмов обработки ошибок.

1. В корневом файле .htaccess имеется закомментированная строка #php_flag error_reporting off, которая, в случае снятия комментария, позволяет отключить показ всех сообщений об ошибках PHP. В этом случае при возникновении сбойной ситуации вы просто увидите белую страницу. Искать ошибку необходимо в логах. По умолчанию строка закомментирована и ошибки отображаются.
2. В файле vars.inc.php имеется строка error_reporting(E_ALL^E_NOTICE), которая рекомендует показывать все ошибки, кроме нотисов. Это вспомогательная строка, она всегда активна, и трогать ее не рекомендуется.
3. В файле vars.inc.php имеется переменная $SHOW_MYSQL_ERRORS, отвечающая за показ ошибок MySQL. Поскольку все запросы MySQL в системе выполняются через специальный класс, то у Вас имеется возможность регулировать возникающие сложности при работе с БД. Если указанный параметр установлен в on, то ошибки отображаются, если в off – то нет.

Независимо от указанного параметра, вы всегда можете отлавливать результаты запроса. Пример:

```
if ($db->captured_errors) $db->vardump($db->captured_errors);
```

Данный код необходимо ставить после нужного запроса. Массив $db->captured_errors содержит результаты выполнения запроса, а в случае ошибки и ее саму.

Для вывода результатов запроса на экран удобно использовать функцию $db->debug(), которая отображает сам запрос и полученные результаты в виде удобной таблицы.

Использование MySQL

Для работы с БД можно использовать непосредственно функции API системы listQuery(), которая описана в конце данного Руководства, или использовать стандартные фунции PHP (mysql_query()); идентификатор соединения в данном случае $LinkID.

Использование вставок PHP-кода

Система NetCat не обременена большим собственным API, чтобы вам было удобно работать с уже известными технологиями, т.е. с PHP.

Вставляемая функция в макетах или компонентах обрамляется кавычкой и точкой. Например:

```
".print("Hello World")."
```

Для вставки более громоздких конструкций рекомендуем использовать следующий механизм:

- В макете дизайна (поле Header или Footer), в компоненте (в Полном выводе объекта):

```
HTML-код
";
PHP-код
echo "
HTML-код
```

- В компоненте (в Префиксе, Объекте в списке, Суффиксе):

```
HTML-код
";
PHP-код
$result .= "
HTML-код
```

Использование ключа подтверждения операций

Иногда нужно защититься от злоумышленников, которые могут вам «подсунуть» каким-либо образом ссылку на вашем сайте, которая производит какое-либо действие (например, удаление — http://сайт/new/drop_news_1.html). Для этого можно использовать ключ подтверждения (token). Ключ

подтверждения уникален для каждого пользователя, другие пользователи его не знают. Ключ подтверждения обычно задается в форме в скрытом поле, либо в ссылке в GET-параметрах.

Следующие методы используются для работы с токенами:

$nc_core->token->is_use ($action)

Возвращает булевое значние - используется ли ключ подтверждения для действия, переданного в качестве параметра или нет.
$action может принимать следующие знчания:

- 'add' — добавление;
- 'change', 'edit', 'message' — изменение;
- 'drop' — удаление.

$nc_core->token->get($user_id = 0)

Возвращает ключ для пользователя с номером $user_id , если $user_id не задан, то берется номер текущего пользователя.

$nc_core->token->get($user_id = 0)

Возвращает html-код со скрытым полем, содержащее значение ключа для пользователя $user_id. Его можно использовать в альтернативных формах, например:

```
".( $nc_core->token->is_use($action) ? $nc_core->token-
>get_input() : "")."
```

В этом случае ключ подтверждения появится если только он реально используется.

$nc_core->token->get_url ($user_id = 0)

Возвращает строку вида «nc_token=ключ», которую можно использовать в ссылках, например:

```
http://example.org/news/drop_news_1.html?'.$nc_core->token-
>get_url()
```

RSS

Безусловно, в NetCat можно реализовать вывод информации в формате RSS. Для этого существуют шаблоны компонента типа «RSS». Подробная информация об этом есть в части «Шаболны компонента».

Часть 10. Система событий

Принцип работы системы событий

Основная идея системы событий заключается в возможности отслеживания базовых действий в системе. Это может быть добавление или изменение сайта, удаление раздела и проч.

Система генерирует событие по факту. Например, удаление раздела — это запрос к базе данных на удаление записи. Это и есть *событие*. При этом все так называемые *слушатели* (подписчики на событие) получат необходимые данные о его совершении. Подписчики назначаются при подключении модуля или библиотеки.

Важно: событие генерируется после обновления системной информации в базе данных. Если событие связано с удалением объекта, будет абсолютно бессмысленно пытаться получить данные об объекте после его удаления.

Система разделяет два типа событий: базовые системные события (те, которые уже содержатся в ядре системы) и пользовательские события (те, которые добавил сам разработчик сайта). Принципиально они ничем не отличаются.

У каждого события есть **имя** (в именах могут использоваться только латинские буквы, цифры и знак подчеркивания). Имена всех системных событий строятся по шаблону:
«действиеСущность».

В качестве действия могут выступать добавление *add*, изменение *update,* включение *checked,* выключение *unchecked,* удаление *drop,* авторизация *authorize.* Сущности описаны в таблице ниже:

Название сущности	**Ключевое слово сущности**
Сайт	Catalogue
Раздел	Subdivision
Компонент в разделе	SubClass
Компонент	Class
Шаблон компонента	ClassTemplate

Сообщение	Message
Системная таблица	SystemTable
Макет дизайна	Template
Пользователь	User
Комментарий	Comment

Таким образом событие *addMessage* возникает при добавление нового объекта, а событие *dropUser* – при удалении пользователя. Далее по тексту будут приведены все системные события.

Так как же работает система событий?

Каждое событие несёт в себе определённый набор параметров, которые могут быть использованы при его отслеживании. Например, при обновлении раздела (событие *updateSubdivision)* передаются параметры *Catalogue_ID* и *Subdivision_ID*, причём именно в указанной последовательности, т. к. в таком виде они попадут в функцию, вызываемую при «улавливании» события.

Ниже будет приведён более подробный пример с кодом.

Трансляция и «улавливание» событий происходит через расширение *event* системного объекта $nc_core. У данного расширения есть два метода, которые нас интересуют — bind() и execute(). Именно они отвечают за отслеживание и трансляцию событий.

Прикрепление событий

Рассмотрим подробно, как же отслеживать событие. Допустим, у нас есть объект $myObject, который инициализируется в каком либо модуле. В конструкторе класса объекта следует написать код:

```
// системный объект
$nc_core = nc_Core::get_object();

// прикрепление события
$nc_core->event->bind($this, "updateSubdivision");
```

В приведённом примере сначала получается системный объект $nc_core, после чего вызывается метод bind(), который привязывает наш объект $myObject к событию *updateSubdivision*. Теперь при трансляции события будет вызван метод нашего объекта, причём название метода должно совпадать с названием

события updateSubdivision(). Это означает, что в классе объекта присутствует метод updateSubdivision().

Важно: в метод объекта будут переданы все параметры, доступные событию, в той последовательности, в которой они описаны в данной документации.

Следует отметить, что привязка события к объекту должна произойти как можно раньше в процессе исполнения кода ядра. Этим и продиктовано стремление разместить привязку непосредственно при инициализации объекта, то есть в его конструкторе. Но можно поступить следующим образом:

```
// системный объект
$nc_core = nc_Core::get_object();

// прикрепление события
$nc_core->event->bind($myObject, "updateSubdivision");
```

В этом случае код может быть написан где угодно в "зоне видимости" объекта $myObject.

Если при улавливании события нужно вызывать метод с названием, отличным от названия события, в методе прикрепления следует передать массив переназначения, это следует сделать во втором параметре:

```
// системный объект
$nc_core = nc_Core::get_object();

// прикрепление события
$nc_core->event->bind( $myObject, array("updateSubdivision" =>
"myMethodRun") );
```

Таким образом, после обновления раздела, наряду с другими слушателями события, будет вызван метод myMethodRun() нашего объекта $myObject с двумя параметрами — *Catalogue_ID* и *Subdivision_ID*.

Важно: последний параметр в транслируемом событии может быть как целым числом (в случае, если в действии участвовал один объект), так и массивом чисел (при групповом действии). Методы-слушатели должны учитывать это.

Трансляция событий

Трансляция событий в системе происходит автоматически. Если же вы пишете свой функционал добавления, обновления или удаления данных из описанных

здесь сущностей — следует позаботиться и о собственной трансляции событий в систему.

Трансляция события происходит следующим образом:

```
// трансляция события
$nc_core->event->execute("updateSubdivision", $CatalogueID,
$SubdivisionID);
```

Важно: в момент трансляции события следует передавать параметры, нужные событию, именно в той последовательности, в которой они описаны в данной документации.

Например, был написан модуль учёта рогов и копыт, который обновляет таблицу раздела. Раздел обновляется, а следовательно, нужно сообщить об этом всем слушателям системы:

```
// обновление раздела
$db->query("UPDATE `Subdivision` SET `Created` = '' WHERE
`Subdivision_ID` = 5");

// трансляция события
$nc_core->event->execute("updateSubdivision", $CatalogueID, 5);
```

Таким образом, обновив раздел, мы сообщим системе о проделанных изменениях.

Важно: последний параметр в транслируемом событии может быть как целым числом (в случае если в действии участвовал один объект), так и массивом чисел (при групповом действии). Методы-слушатели должны учитывать это.

Пользовательские события

Помимо системных событий вы можете ввести свои пользовательские события. Для этого, как и прежде, используется расширение **event**. Метод, регистрирующий новое событие, называется **register_event** и требует два параметра — имя события и его описание.
Пример:

```
// системный объект
$nc_core = nc_Core::get_object();

// создание нового события
$nc_core->event->register_event("my_event", "Мое событие");
```

Имя события должно содержать только латинские буквы, цифры и знак подчеркивания. Если событие будет создано, то метод **register_event** вернет **true**, иначе - **false**.

Список системных событий

В таблице приведены все системные события.

Событие	Описание	Параметры
addCatalogue	добавление сайта	Catalogue_ID
updateCatalogue	обновление сайта	Catalogue_ID
dropCatalogue	удаление сайта	Catalogue_ID
checkCatalogue	включение сайта	Catalogue_ID
uncheckCatalogue	выключение сайта	Catalogue_ID
addSubdivision	добавление раздела	Catalogue_ID, Subdivision_ID
updateSubdivision	обновление раздела	Catalogue_ID, Subdivision_ID
dropSubdivision	удаление раздела	Catalogue_ID, Subdivision_ID
checkSubdivision	включение раздела	Catalogue_ID, Subdivision_ID
uncheckSubdivision	выключение раздела	Catalogue_ID, Subdivision_ID
addSubClass	добавление компонента раздела	Catalogue_ID, Subdivision_ID, Sub_Class_ID
updateSubClass	обновление компонента в разделе	Catalogue_ID, Subdivision_ID, Sub_Class_ID
dropSubClass	удаление компонента в разделе	Catalogue_ID, Subdivision_ID, Sub_Class_ID
checkSubClass	включение компонента в разделе	Catalogue_ID, Subdivision_ID, Sub_Class_ID

uncheckSubClass	выключение компонента в разделе	Catalogue_ID, Subdivision_ID, Sub_Class_ID
addMessages	добавление сообщения	Catalogue_ID, Subdivision_ID, Sub_Class_ID, Class_ID, Message_ID
updateMessages	обновление сообщения	Catalogue_ID, Subdivision_ID, Sub_Class_ID, Class_ID, Message_ID
dropMessages	удаление сообщения	Catalogue_ID, Subdivision_ID, Sub_Class_ID, Class_ID, Message_ID
checkMessage	включение объекта	Catalogue_ID, Subdivision_ID, Sub_Class_ID, Class_ID, Message_ID
uncheckMessage	выключение объекта	Catalogue_ID, Subdivision_ID, Sub_Class_ID, Class_ID, Message_ID
addClass	добавление компонента	Class_ID
updateClass	обновление компонента	Class_ID
dropClass	удаление компонента	Class_ID
addClassTemplate	добавление шаблона компонента	Class_ID (чей шаблон), Class_ID (шаблон)
updateClassTemplate	обновление шаблона компонента	Class_ID (чей шаблон), Class_ID (шаблон)
dropClassTemplate	удаление шаблона	Class_ID (чей

	компонента	шаблон), Class_ID (шаблон)
addTemplate	добавление макета дизайна	Template_ID
updateTemplate	обновление макета дизайна	Template_ID
dropTemplate	удаление макета дизайна	Template_ID
updateSystemTable	обновление системной таблицы	System_Table_ID
addUser	добавление пользователя	User_ID
updateUser	обновление пользователя	User_ID
dropUser	удаление пользователя	User_ID
checkUser	включение пользователя	User_ID
uncheckUser	выключение пользователя	User_ID
authorizeUser	авторизация пользователя	User_ID
addComment	добавление комментария	Catalogue_ID, Subdivision_ID, Sub_Class_ID, Class_ID, Message_ID, Comment_ID
updateComment	обновление комментария	Catalogue_ID, Subdivision_ID, Sub_Class_ID, Class_ID, Message_ID, Comment_ID
dropComment	удаление комментария	Catalogue_ID, Subdivision_ID, Sub_Class_ID, Class_ID, Message_ID, Comment_ID

Параметры данной таблицы означают:

Catalogue_ID – номер сайта;
Subdivision_ID — номер раздела;

Sub_Class_ID — номер компонента в разделе;
Class_ID — номер компонента;
Message_ID — номер сообщения (объекта);
System_Table_ID — номер системной таблицы;
User_ID — номер пользователя;
Comment_ID — номер комментария.

Пример

Приведем полноценный пример использования системы событий.
Допустим, мы хотим отслеживать авторизацию пользователей из группы «Администрация» (группа имеет номер 1). По итогам администратор будет получать письмо с уведомлением о каждой авторизации пользователя.
Для отслеживания события *authorizeUser (*авторизация пользователя) введём класс ListenUser:

```
class ListenUser {
     public function __construct () {
          $nc_core = nc_Core::get_object();
          $nc_core->event->bind($this, array('authorizeUser' =>
'authorize_user') );
     }
}
```

У нашего класса есть конструктор, в котором две строчки: на первой строчке идет получение системного объекта *$nc_core*, который содержит расширение *event*, нужное для работы с событиями;
на второй строчке конструктора идет привязывание события *authorizeUser* к объекту класса *ListenUser*. При трансляции события *authorizeUser* будет вызван метод *authorize_user* нашего класса. Теперь приведем сам метод *authorize_user*, который узнает номер группы пользователя и, в случае необходимости, отправляет письмо:

```
public function  authorize_user ( $user_id ) {
     $nc_core = nc_Core::get_object();
     $system_env = $nc_core->get_settings();

     $groups = nc_usergroup_get_group_by_user($user_id);

     if ( in_array( 1, $groups ) ) {
          $mailer = new CMIMEMail();
          $mailer->mailbody('Авторизация пользователя '.$user_id);
          $mailer->send( 'admin@example.com',
                         $system_env['SpamFromEmail'],
                         $system_env['SpamFromEmail'],
                         'Авторизация пользователя',
                         $system_env['SpamFromName']);
```

```
	}

	return 0;
}
```

В начале работы этой функции мы получаем системный объект $nc_core, далее с помощью метода get_settings получаем настройки системы и записываем их в массив $system_env. Этот массив содержит такие элементы, как:
$system_env['SpamFromEmail'] — адрес, откуда отправляются письма
и
$system_env['SpamFromName'] — имя, от которого отправляются письма.
Эти параметры задаются в настройках системы (меню: «Настройки» — «Настройки системы»)

С помощью функции nc_usergroup_get_group_by_user можно получить номера всех групп, в которых состоит пользователь. Описание этой функции можно найти в главе «Функции для работы с группами пользователей» данного руководства.

После того, как в массив $groups мы записали номера всех групп авторизированного пользователя, нужно проверить, входит ли в них группа с номером 1. Для этого используется php-функция in_array. Если нужная группа входит в их число – отправляем письмо с помощью класса CMIMEMail. Данный класс также описан в данном руководстве, так что останавливаться на нем не будем, отметим только то, что письмо отправится на адрес admin@example.com.

После того, как класс был объявлен, надо создать его экземпляр:

```
$listenObj = new  ListenUser();
```

Весь код можно поместить в модуль «Интерфейс разработчика», в файл function.inc.php
Этот файл будет выглядеть так:

```
<?php

class ListenUser {
	public function __construct () {
		$nc_core = nc_Core::get_object();
		$nc_core->event->bind($this, array('authorizeUser' =>
'authorize_user') );
	}

	public function  authorize_user ( $user_id ) {
		$nc_core = nc_Core::get_object();
```

```
            $system_env = $nc_core->get_settings();

            $groups = nc_usergroup_get_group_by_user($user_id);

            if ( in_array( 1, $groups ) ) {
                 $mailer = new CMIMEMail();
                 $mailer->mailbody('Авторизация пользователя '.$user_id);
                 $mailer->send( 'admin@example.com',
                                $system_env['SpamFromEmail'],
                                $system_env['SpamFromEmail'],
                                'Авторизация пользователя',
                                $system_env['SpamFromName']);
            }

            return 0;
       }
}

$listenObj = new  ListenUser();

?>
```

Теперь, после авторизации пользователя из группы «Администрация», администратору сайта будет отправлено письмо-уведомление на адрес admin@example.com.

Часть 11. Модернизация системы

Установка модулей

Для установки любого модуля требуется на время установки поставить права на папки /netcat/tmp и /netcat/modules в 777 или же установить у них принадлежность к тому же пользователю ОС на сервере, к которому принадлежит apache.

Установка модуля осуществляется через веб-интерфейс. Вам необходимо зайти в режиме администрирования в раздел «Инструменты» - «Модули» и закачать дистрибутив модуля (в виде TGZ-архивов).

После корректной установки в разделе Модули появится соответствующая запись.

После успешной установки для некоторых модулей может потребоваться дополнительное ее завершение, а именно создание специальных разделов для полноценной работы модуля. В этом случае в списке напротив установленного модуля будет стоять ссылка «завершить установку». Откройте данную ссылку и следуйте дальнейшим инструкциям.

Установка обновлений системы

Обновления системы (патчи) решают задачи исправления найденных недочетов в системе или обновления ее до новых версий. Новые патчи выкладываются на сайте netcat.ru по мере их появления. Интерфейс обновления системы идентичен интерфейсу установки модулей. В случае неудачной установки патча вы получите соответствующее сообщение в браузере.

Перед установкой патча необходимо на все файлы и папки системы поставить права 777. Обычно это описано в инструкции к патчу, там же указано, как это сделать. Обычно для этого нужно выполнить команду в консоли (по SSH или попросить администратора сервера):

```
chmod  -R 777 ./папка
папка - директория, где лежат все файлы системы
```

После корректной установки в разделе «Обновление системы» появится соответствующая запись.

Часть 12. Разработка модулей

Структура модуля

Ниже приведен список файлов и директорий, которые содержит установочный архив модуля. Серым цветом помечены файлы, присутствие которых в инсталляторе обязательно.

/netcat/	папка содержит дополнительные файлы необходимые для работы модуля, перечисленные в файле files.txt
admin.php	основной файл управления модулем из административной зоны
admin.inc.php	файл с функциями используемыми файлом admin.php
en.lang.php	языковой файл, содержащий все текстовые константы, используемые для обеспечения мультиязычная
files.txt	список файлов, входящих в модуль помимо обязательных
function.inc.php	функции, используемые модулем
id.txt	основная информация о модуле и совместимости версий
index.php	индексный скрипт модуля, может быть пустым (<?php ?>)
install.php	вспомогательный инсталляционный файл модуля
message.txt	зарезервировано
message_int.txt	зарезервировано, мультиязычная поддержка
parameters.txt	параметры (константы) модуля, отображаемые в настройках
ru.lang.php	языковой файл, содержащий все текстовые константы
sql.txt	запросы для БД, выполняемые при установке (формат MySQL)
sql_int.txt	запросы для БД, выполняемые при установке (формат MySQL), мультиязычная поддержка
url_routes.js	системный файл предназначенный для расширения визуальных возможностей управления модулем. Его следует поместить в папку /netcat/modules/имя_модуля/ и описать в файле files.txt
ui_config.php	используется совместно с файлом url_routes.js, описание будет приведено ниже по тексту.
setup.php	файл содержит набор действий для настройки после установки, это требуется, когда например нужно создать новый раздел под нужды модуля.

Подробное описание файлов:

id.txt

Данный файл содержит основную информацию о модуле в следующем формате (построчно):

1. ключевое слово модуля (строчными латинскими буквами без пробелов)

2. код названия системы («2» для Standard, «3» для Extra, «4» для Community, «6» для E-Commerce, «7» для SEO, «8» для Corporate)
3. версия системы NetCat, для которой изготовлен модуль (например, «3.6»)
4. номер патча, который должен быть установлен (например, строка «352» означает, что перед установкой модуля необходимо установить обновление с номером «352»)
5. название модуля на русском языке, обычно заменяется названием константы из языкового файла, например: NETCAT_MODULE_BLOG.
6. зарезервировано
7. зарезервировано
8. описание модуля, обычно заменяется названием константы из языкового файла модуля, например: NETCAT_MODULE_BLOG_DESCRIPTION.

files.txt

Данный файл содержит список файлов, которые входят в модуль помимо обязательных. Файлы указываются с полным путем от корня сайта. Например, если мы хотим, чтобы при установке в папку модуля "mymod" записался в файл test.php, то нам необходимо записать в files.txt следующую строку

./netcat/modules/mymod/test.php

При этом сам файл test.php должен находиться в архиве модуля в аналогичной папке - /netcat/modules/mymod/.

parameters.txt

Файл содержит список констант модуля и их значения. На каждой строчке указывается по одной константе. При загрузке страниц сайта константы загружаются в двумерный массив $MODULE_VARS, первым индексом которого является ключевое слово модуля, а вторым – название константы. После установки модуля содержимое этого файла записывается в поле Parameters таблицы Module, а сам файл удаляется. В процессе эксплуатации модуля значения параметров (констант) можно изменять в режиме администрирования в разделе «Инструменты» - «Модули».

Например, указываем в настройках модуля с ключевым словом "mymod":

```
MY_VAR=123
```

В результате, при работе с компонентами, макетами дизайна или скриптами модулей, мы получаем элемент MY_VAR в массиве $MODULE_VARS[mymod] - **$MODULE_VARS[mymod][MY_VAR]**. Этому элементу будет присвоено значение «**123**».

sql.txt

Данный файл содержит список SQL-запросов для БД, запускаемых при установке модуля. Каждая строка содержит по одному SQL-запросу.

install.php

Данный файл содержит две функции, контролирующие установку модуля.

Первая функция **CheckAbilityOfInstallation()** активизируется до установки модуля и проверяет, возможна ли эта операция. По умолчанию функция возвращает положительный результат. Вы можете модифицировать эту функцию, поставив туда проверку на все необходимые условия, и, если по каким-либо причинам система не удовлетворяет условиям установки модуля, Вы можете установить значение $result[Success] в 0, при этом указав сообщение об ошибке в $result[ErrorMessage]. В этом случае установка модуля произведена не будет.

Вторая функция **InstallThisModule()** производит действия, необходимые для установки модуля, помимо стандартных (распаковка архива, создание и запись в папку с модулем, посыл запросов в БД, создание соответствующей строчки в таблице Module). Вы можете дописать в эту функцию код, необходимый для установки Вашего модуля; если установка не была произведена корректно, установите значение $result[Success] в 0, при этом указав сообщение об ошибке в $result[ErrorMessage]. В этом случае установка модуля произведена не будет.

В функции **InstallThisModule()**, в частности, можно произвести импорт компонента (TPL-файла). Для этого необходимо записать в архив модуля файл компонента (*.tpl) и воспользоваться функцией **ParseClassFile($file)**, где $file – полный путь к TPL-файлу. В качестве результата функция возвратит номер созданного компонента. Для генерации TPL-файла воспользуйтесь функцией экспорта компонента.

index.php и **function.inc.php**

Данные файлы являются основными скриптами модуля. Индексный файл производит необходимые действия, связанные с работой модуля, используя функции из файла function.inc.php, который подключается автоматически. Если в каком-либо из этих файлов нет необходимости, то они могут быть пустыми. Наличие каждого файла обязательно.

ru.lang.php

Файлы данного типа содержат все текстовые константы, используемые в модуле для обеспечения мультиязычности. Если Ваш сайт, скажем, имеет еще и немецкий интерфейс, Вам необходимо создать файл ger.lang.php (в папке / netcat/admin/lang/ должен присутствовать файл ger.php).

Пример содержимого файла ger.lang.php:
define("NETCAT_MODULE_AUTH", "Интерфейс пользователя");
define("NETCAT_MODULE_AUTH_DESCRIPTION", "Интерфейс пользователя в системе ввода-вывода. Возможность регистрации внешней группы пользователей, изменение собственной анкеты, пароля, восстановление пароля. Данный модуль может интегрироваться с другими модулями системы.");

url_routes.js

По умолчанию файл должен присутствовать, он должен содержать следующие строки:

```
urlDispatcher.addRoutes( {

  1: '' // dummy entry

} );
```

Если в модуле есть управляющий интерфейс (файл admin.php), то добавляется ещё одна строка с путём:

```
urlDispatcher.addRoutes( {

   'module.имя_модуля': NETCAT_PATH+'modules/имя_модуля/admin.php',
  1: '' // dummy entry

} );
```

Обратите внимание, на значение «имя_модуля», вместо этого следует вписать ключевое слово модуля.

Описание возможностей описано ниже, в пункте «Элементы управления».

Процесс написания модуля

В процессе написания модуля нет необходимости в наличии всех файлов, достаточно только обязательных – они нужны для того, чтобы подготовить модуль к установке, т.е. для создания установочного архива. Если модуль создается для использования в рамках конкретного проекта, подготовки к установке не требуется. По умолчанию в системе предустановлен пустой модуль «Интерфейс разработки» (ключевое слово - «default»), который Вы можете использовать для подключения собственных скриптов и функций.

Перед написанием нового модуля необходимо произвести следующие действия:

1. Создать папку **/netcat/modules/ключевое_слово/**
2. Создать в этой папке файл **function.inc.php**
3. Добавить строку с информацией о модуле в таблицу **Module:**
INSERT INTO Module (Module_Name,Keyword,Description)
VALUES
(‘MODULE_NAME','ключевое_слово','MODULE_DESCRIPTION');
Здесь MODULE_NAME и MODULE_DESCRIPTON – константы, которые необходимо создать во всех языковых файлах из папки /netcat/admin/lang/ (про систему мультиязычности читайте одноименную главу в данном Руководстве)

После этого Вы можете описывать функции в файле function.inc.php (они будут автоматически подключаться системой), создавать дополнительные скрипты в папке модуля, использовать и объявлять константы **$MODULE_VARS[ключевое_слово]**.

Элементы управления

Чтобы внутри административной зоны можно было использовать стилизованные под системные, свои собственные элементы управления, например, кнопку «подтвердить», в начале файла admin.php, наряду с другими включениями, следует также прописать следующие строки, обозначенные жирным шрифтом:

```
require_once "../../vars.inc.php";
```

```
require "admin.inc.php";
require "function.inc.php";
require $ADMIN_FOLDER."function.inc.php";
require_once $INCLUDE_FOLDER.'s_loadenv.inc.php';

require $ADMIN_FOLDER."modules/ui.php";
require_once("ui_config.php");
$UI_CONFIG = new ui_config_module_calendar('admin');

if ( is_file($MODULE_FOLDER.'blog/'.MAIN_LANG.'.lang.php') )
        require_once($MODULE_FOLDER.'blog/'.MAIN_LANG.'.lang.php');
else
        require_once($MODULE_FOLDER.'blog/en.lang.php');

LoadModuleEnv();
```

Обратите внимание, что в названии класса присутствует ключевое слово модуля «calendar».

Если нужно выделить управляющую кнопку, в код формы стоит вместо стандартной кнопки подтверждения, следует прописать управляющую команду:

```
$UI_CONFIG->actionButtons[] = array(
        "id" => "submit",
        "caption" => "подтвердить",
        "action" => "mainView.submitIframeForm('mainForm')"
);
```

При необходимости, в этом же массиве можно указать параметр выравнивания элемента:

```
$UI_CONFIG->actionButtons[] = array(
        "id" => "submit",
        "caption" => "подтвердить",
        "action" => "mainView.submitIframeForm('mainForm')",
        "align" => "left"
);
```

Соответственно, у формы нужно прописать идентификатор:

```
<form method='post' id='mainForm' action='admin.php'>
...
$UI_CONFIG->actionButtons[] = array(
        "id" => "submit",
        "caption" => "подтвердить",
        "action" => "mainView.submitIframeForm('mainForm')"
);
```

```
...
</form>
```

В таком случае, кнопка «подтвердить» будет расположена на нижней панельке системы управления. Следует заметить, значение переменной $UI_CONFIG должно быть объявлено глобально:

```
global $UI_CONFIG;
```

Для организации закладок, в интерфейсе управления модулем, следует модифицировать файлы ui_config.php и url_routes.js

Допустим нужно расширить стандартный модуль «Календарь», установив в его интерфейсе управления несколько вкладок для разных задач. Первая вкладка будет выводить администратора на стандартные настройки календаря, а вторая будет писать приветствие. Для организации такой возможности следует изменить содержимое файлов ui_config.php и url_routes.js, пример приведён ниже.

Содержимое файла ui_config.php, задаёт нужные вкладки:

```
<?php

class ui_config_module_calendar extends ui_config_module
{

# $toolbar_action = 'main' - определяет действие по-умолчанию
function ui_config_module_calendar($active_tab = 'admin', $toolbar_action = 'main')
{
     global $db, $MODULE_FOLDER;

     $this->ui_config_module('calendar', $active_tab);

     if ($active_tab == 'admin') {

    $this->toolbar[] = array('id' => "main",
                             'caption' => 'Main',
                             'location' => "module.calendar.main",
                             'group' => "grp1");

    $this->toolbar[] = array('id' => "test",
                             'caption' => 'Test',
                             'location' => "module.calendar.test",
                             'group' => "grp1");

     $this->locationHash = "module.calendar.$toolbar_action";
     $this->activeToolbarButtons[] = $toolbar_action;
    }
}
```

```
}

?>
```

Содержимое файла url_routes.js, задаёт переходы по вкладкам:

```
urlDispatcher.addRoutes( {

  'module.calendar.main': NETCAT_PATH + 'modules/calendar/admin.php',
  'module.calendar.test': NETCAT_PATH + 'modules/calendar/test.php',

  1: '' // dummy entry
} );
```

Теперь в каталоге модуля нужно создать файл test.php, который содержит незамысловатый код:

```
<?php

echo "Hello world!!!";

?>
```

Теперь интерфейс управления модулем «Календарь», располагает двумя вкладками. При нажатии на вторую вкладку появится сообщение «Hello world!!!»

Подготовка установочного архива

Для создания установочного архива модуля необходимо упаковать все файлы модуля, в т.ч. **обязательные файлы**, в архив. Список обязательных файлов приведен в разделе «Структура модуля». Для упаковки модуля нужно использовать архиватор поддерживающий формат «tar».

Упаковка модуля архиватором tar происходит следующим образом:

```
tar zcvf module_name.tgz *
```

Рекомендуется файлу архива присваивать следующее название:

Производитель_КлючевоеСлово_Версия_ЯдроСистемы.Расширение

Производитель – название компании-производителя модуля (латинскими буквами)
КлючевоеСлово – ключевое слово модуля

Версия – версия модуля
ЯдроСистемы – версия ядра системы
Расширение – tgz

Пример:
aist_auth_1_2.tgz

В ходе установки модуля посредством интерфейса системы будут произведены следующие действия:

1. Распаковка архива в папку **/netcat/tmp/**
2. Создание папки **/netcat/modules/ключевое_слово_модуля/**
3. Копирование файлов в папку модуля
4. Активация запросов для БД
5. Запись в таблицу Module информации об установленном модуле
6. Удаление временных файлов из **/netcat/tmp/**

Часть 13. Инструменты разработчика

Описание инструментов

Предисловие

В процессе разработки сайтов на CMS NetCat нам, как и многим разработчикам приходила мысль о том, что хотелось бы облегчить процесс разработки компонентов и макетов сайта. Сделать его более удобным и функциональным. Для этого мы решили добавить в NetCat дополнительные инструменты, которые призваны облегчить труд разработчиков.

Подсветка синтаксиса

Данный инструмент позволяет переключить скучную textarea в достаточно функциональный редактор текста с подсветкой синтаксиса php,html,css,sql. Данную возможность можно включить для каждого отдельного поля ввода (Префикс объектов в списке,Объект в списке,Системные настройки,Макеты,Альтернативные формы)

Синтаксическая проверка

Часто при разработке и редактировании компонентов и макетов разработчик допускает синтаксические ошибки, чаще всего они связаны с экранированием или «разрывами» макетов и компонентов. После этого разработчик сохраняет свои данные, заходит на сайт и встречает ошибки похожие на ***Parse error: syntax error, unexpected '>' in /var/www/site/netcat/require/s_list.inc.php(570) : eval()'d code on line 2.***
Что говорит о невозможности выполнения кода компонента (макета) в eval'e. Для того чтобы этого можно было избежать еще во время разработки мы сделали инструмент синтаксической проверки. Этот инструмент позволяет, не сохраняя данные, проверить их. По сути, все поля редактируемого в данный момент макета или компонента тоже выполняются в eval — и потом пользователю возвращается результат.

К сожалению, данный инструмент имеет свои ограничения, связанные с особенностями интерпретатора php, и не может 100%-но избавить разработчика от всех допущенных ошибок. Об ограничениях этого инструмента речь пойдет в следующем разделе.

Предпросмотр

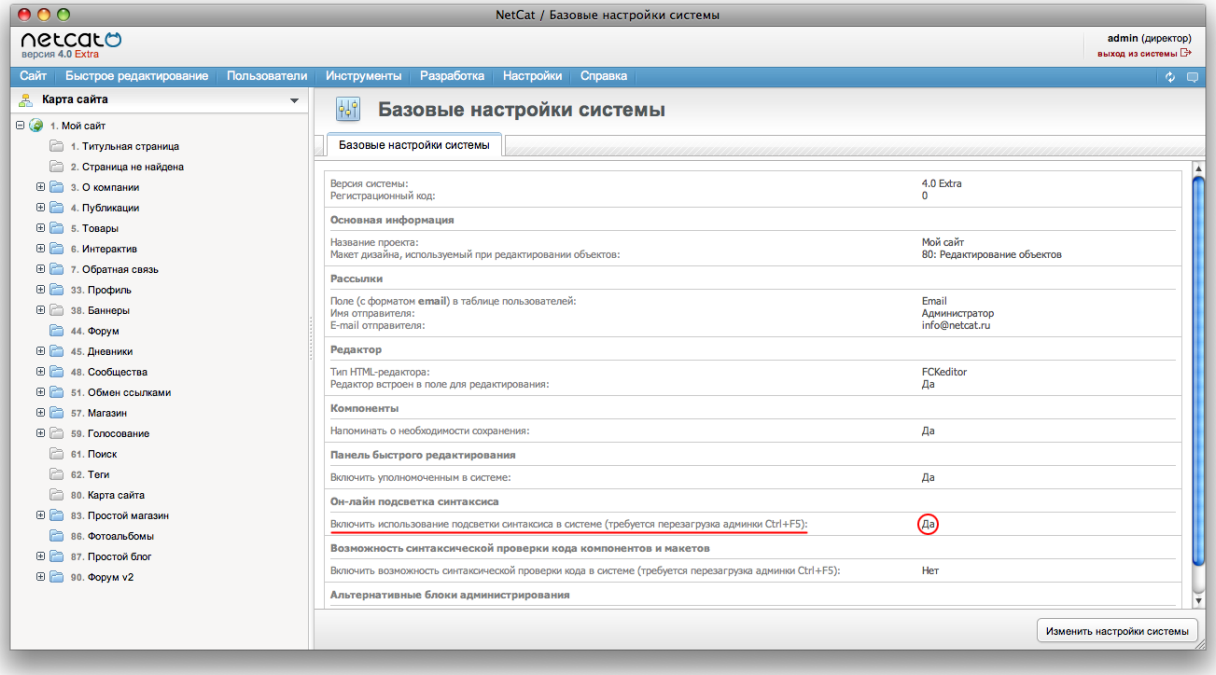
Смысл данного инструмента для разработчика — возможность применить разрабатываемый или изменяемый компонент в «живом» виде без сохранения, в базу данных сайта. Возможная область применения — изменения на рабочем проекте , когда нужно что-то изменить, но совершенно не хочется,чтобы пользователи видели процесс отладки компонента или макета, с возможными ошибками.

Способы работы с этим инструментом, будут описаны в соответствующем пункте следующего раздела.

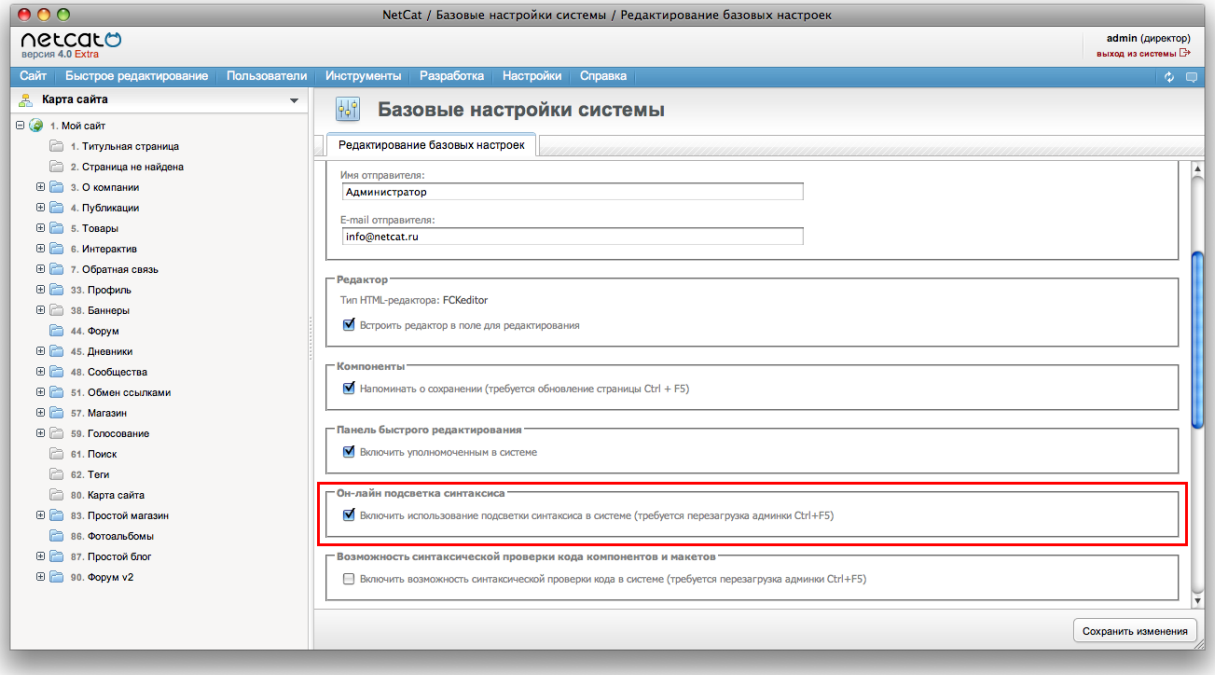
Административная часть

Подсветка синтаксиса

Включение синтаксической подсветки в системе



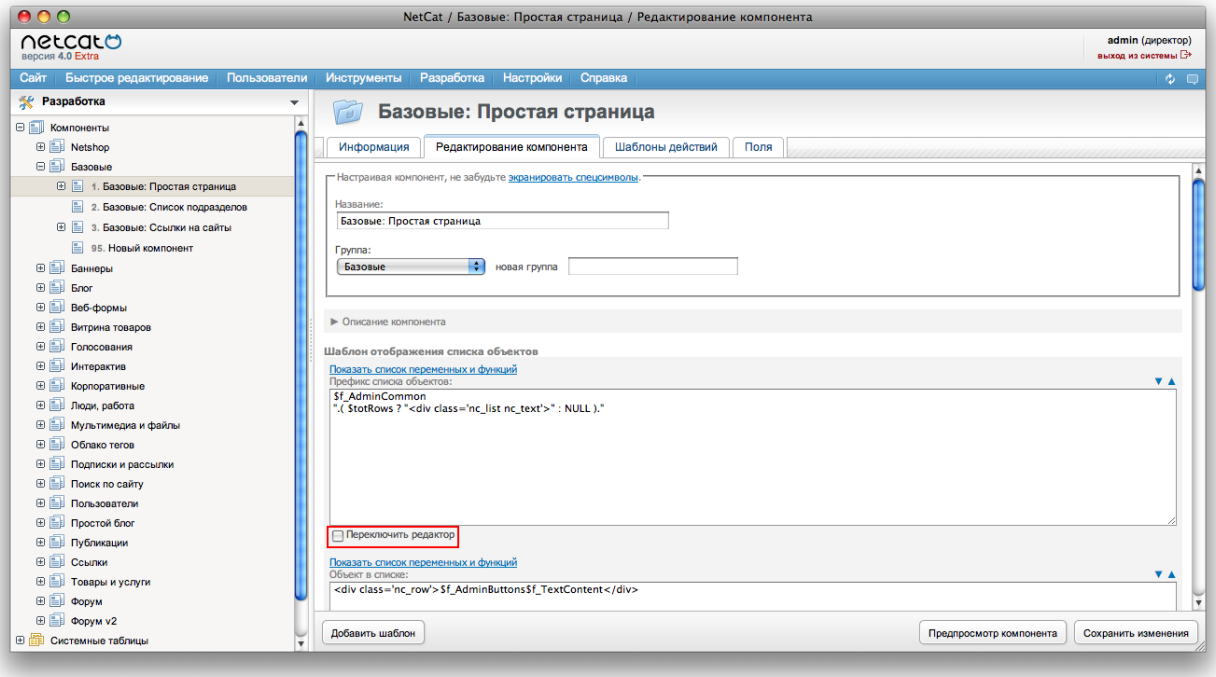
Соответствующий пункт для активации этого инструмента находится в «Настройки» - «Настройки системы» - «Изменить настройки системы» - «Онлайн подсветка синтаксиса».



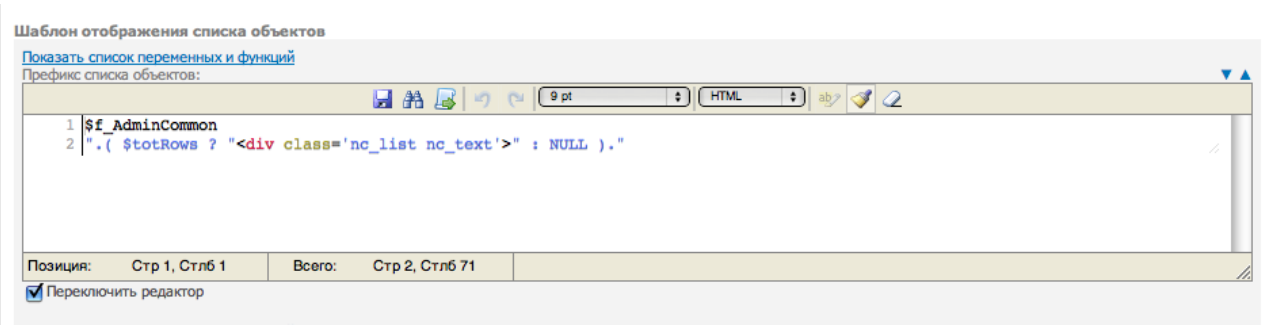
После сохранения настроек системы, нужно перезагрузить админку по Ctrl+F5.

Использование синтаксической подсветки

Открыв окно редактирования компонентов (все то же самое действительно и для макетов) у каждого поля ввода появится переключатель с надписью «Переключить редактор»:



Если нажать на него. То данное поле ввода изменится и примет вид:



Таким образом, будет активирован редактор с подсветкой синтаксиса. В таком режиме поддерживаются все основные функции системы: сохранние данных, синтаксическая отладка и пр.

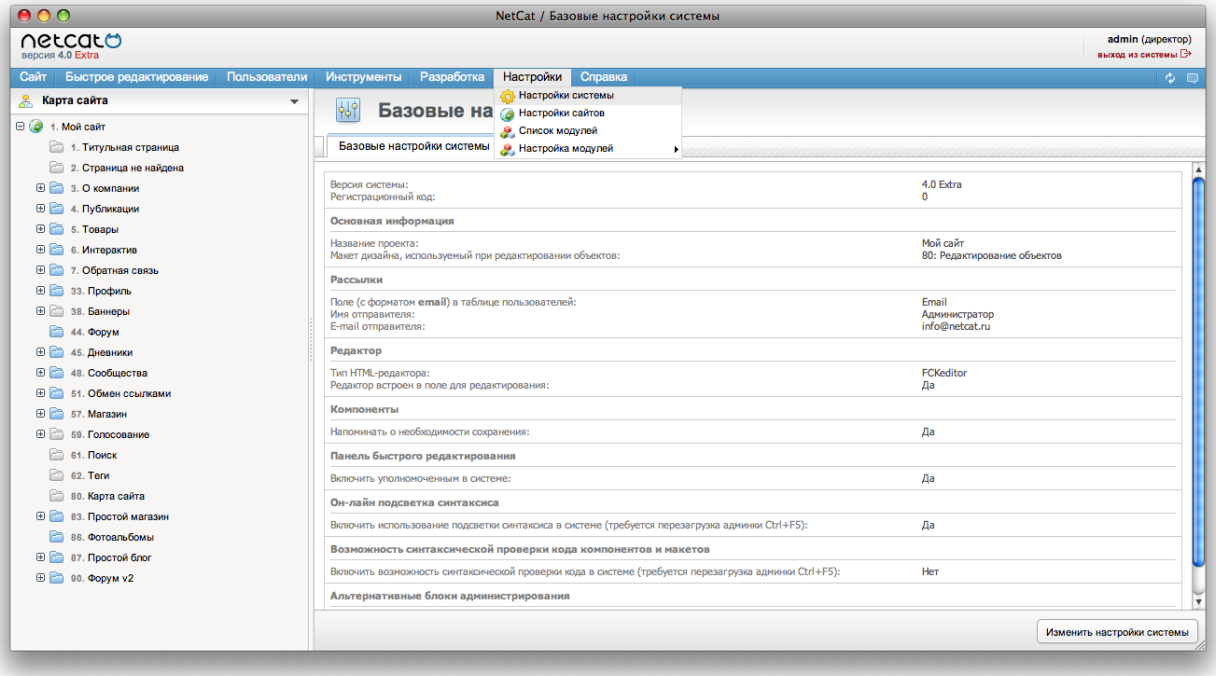
Ограничения редактора

Так как данный инструмент активно использует в своей работе Javascript, то это накладывает определенные ограничения на работу с ним: на больших текстах скорость работы может существенно падать, крайне не рекомендуется к работе в браузерах InternetExplorer 6 и 7 — ввиду особенностей данного браузера при работе с Javascript (особенно низкая производительность)

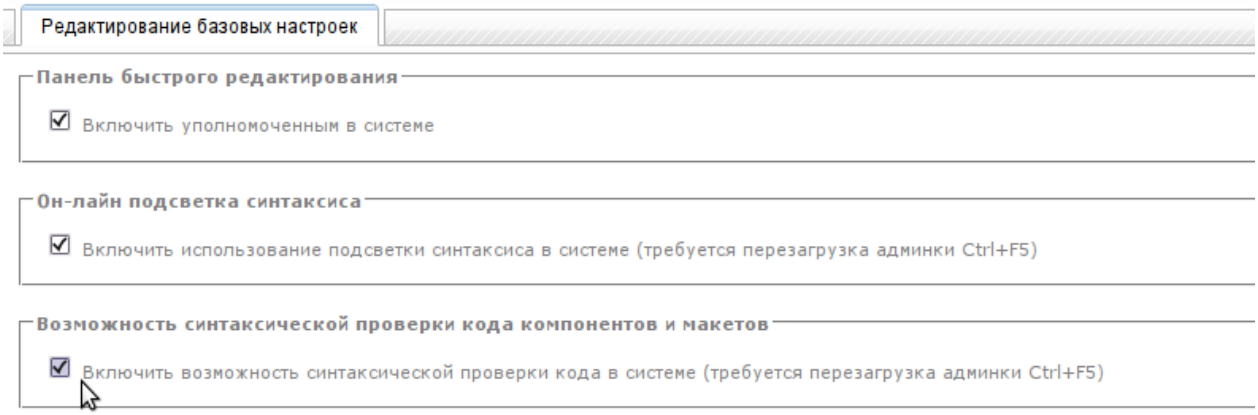
Синтаксическая отладка

Подключение данной возможности

После установки системы, данная возможность отключена по умолчанию, для ее использования необходимо ее, прежде всего, включить. Это возможно сделать на странице «Настройки системы», выбрав в основном меню пункт «Настройки», затем в открывшемся меню «Настройки системы»



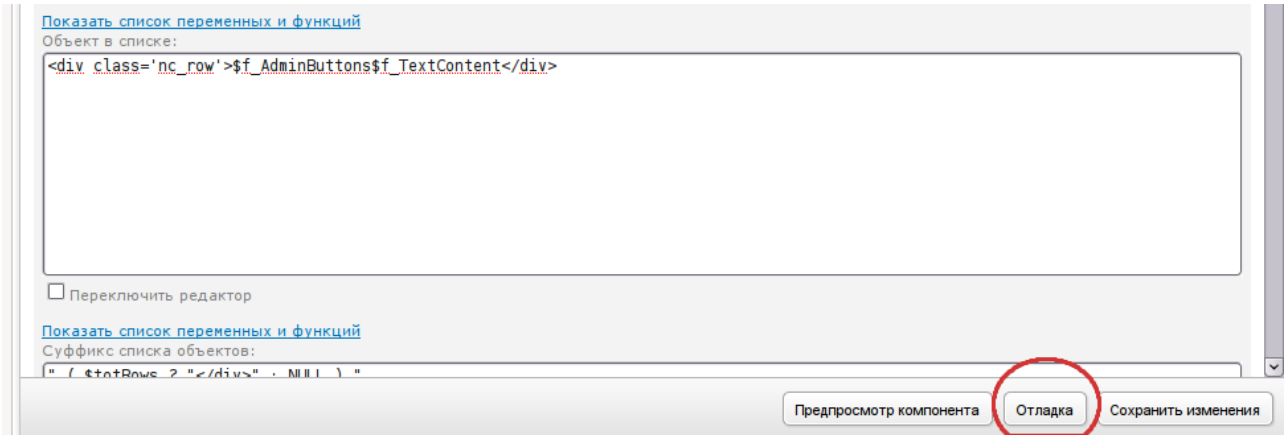
В открывшемся окне настроек системы, нужно нажать на кнопку «Изменить настройки системы». В открывшемся окне редактирования настроек системы нужно поставить галочку в поле «Включить возможность синтаксической проверки ...»



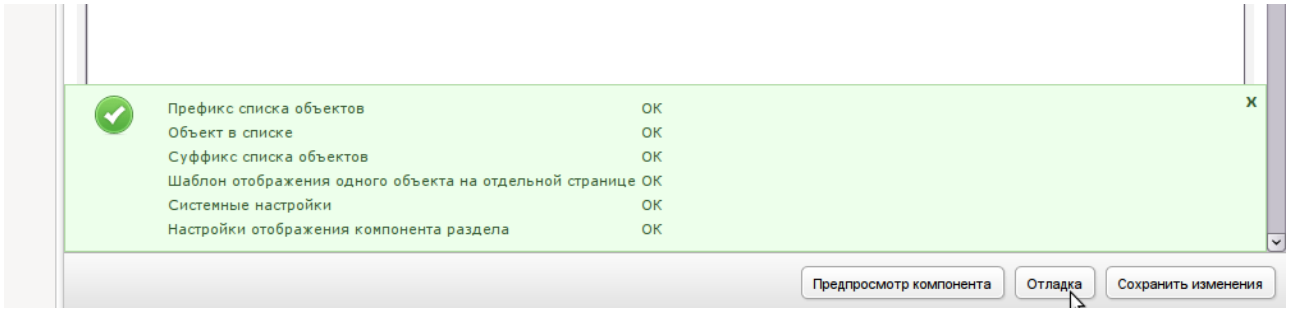
Необходимо сохранить настройки системы, чтобы это изменение имело действие. Теперь возможность проверки доступна в редактировании компонентов (макетов).

Использование

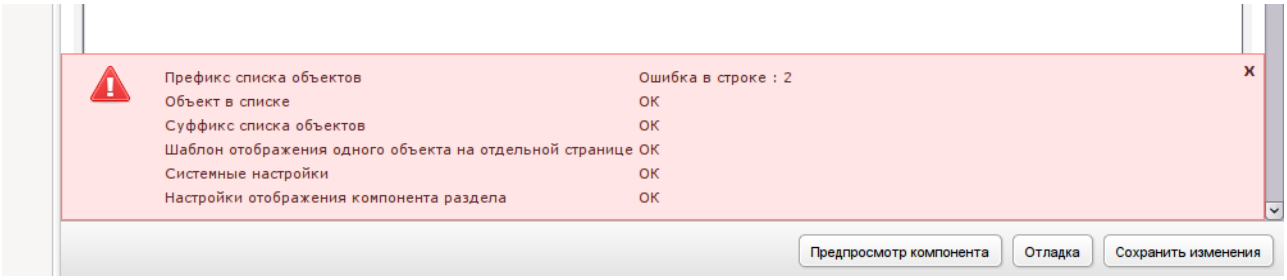
В окне редактирования компонента (все тоже действительно и для макетов), внизу на панели появилась новая функциональная кнопка «Отладка».



При нажатии на нее весь компонент (справедливо и для макетов дизайна) будет отправлен на сервер для проверки. В случае если все успешно разработчик увидит следующее окно.



В случае если где-либо есть ошибка, результат будем немного иным.



Ограничения синтаксической отладки

К сожалению, для данного инструмента существуют ограничения накладываемые особенностями отображения ошибок интерпретатором языка php.

Рассмотрим следующий пример. Допустим в «Префиксе» компонента находится следующий программный код:

```
$f_AdminCommon"
$result.="<h1>$INCLUDE_FOLDER</h1>
".( $totRows ? "<div class='nc_list nc_text'>" : NULL ).
```

Синтаксическая проверка покажет, что ошибка в строке 2 . Хотя на самом деле в первой строке пропущена точка с запятой после двойных кавычек. Интерпретатор языка php встречает $ (знак доллара) что по его мнению ошибочно. Поэтому он считает что ошибка во второй строке, хотя причина ошибки, находится в первой строке.

Рассмотрим еще один пример. Допустим в поле компонента «Объект в списке » находится следующий программный код:

```
";
if ($f_RowID > 12) {
     some_strange_function();
}
$result.=" <div>$f_RowId</div>
```

Синтаксическая проверка такого участка покажет, что все нормально , а во время выполнения, когда будет существовать переменная **$f_RowId** и она станет больше 12 , интерпретатор php выдаст фатальную ошибку потому что не существует такой функции **some_strange_function.** Такую ошибку невозможно отловить на этапе проверки. Так как проверяемые таким образом компоненты (макеты) не выполняются в контексте ядра системы NetCat и все системные переменные (обычно доступные в компонентах или макетах) не определены.

Накладывается также ограничение на использование некоторых php функций в проверяемом коде. Это функции **eval(), ob_end_clean(), header ('Location: some_url ')**. При проверке такого кода с помощью инструмента синтаксической проверки результаты не предсказуемы.

Принципиальная возможность работы

К сожалению, данная возможность синтаксической проверки не совместима с некоторыми версиями php. Мы производили тестирования на различных версиях php и на различных операционных системах. Для принципиальной возможности работы этого инструмента необходимо, чтобы при установленных на вашем сервере разработки настройках php (display errors on и error_reporting 6141 для PHP5 это все ошибки без E_WARNING) установленных в php.ini или .htaccess сервер возвращал код HTTP 200 Ok, а не 500 Internal Server Error. Проверить это можно создав любой ошибочный скрипт

на сервере, и обратившись к нему посмотреть возвращаемый HTTP ответ. Например, с помощью расширения Mozilla FireFox — Live HTTP Headers. Если ваш сервер возвращает 500 — то возможность синтаксической отладки у вас работать не будет.

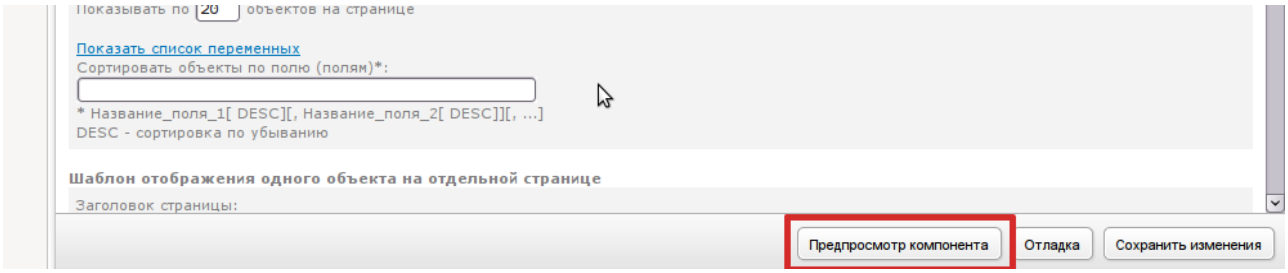
Резюме

Используя этот инструмент — вы должны четко понимать что происходит. Именно поэтому из-за накладываемых ограничений, по умолчанию такая возможность в системе отключена.

Предпросмотр

Использование

В окне редактирования компонента (все тоже действительно и для макетов), внизу на панели появилась новая функциональная кнопка «Предпросмотр ...».



Нажав на эту кнопку, данные из формы редактирования компонента (справедливо для макетов дизайна,альтернативных форм) будут переданы для формирования компонента в режиме предпросмотра.

Особенности работы

Для того чтобы компоненты можно было просматривать в режиме предпросмотра отображения, необходимо чтобы такой компонент был добавлен хотя бы в один раздел. Если такого компонента не будет , вы получите предупреждающее сообщение о том, что необходимо добавить такой компонент хотя бы в один раздел. Компонент должен быть добавлен с действием по умолчанию - «Просмотр».

Для предпросмотра альтернативных форм добавления и редактирования — компоненты должны быть добавлены в раздел. В случае с альтернативной формой изменения в компоненте в разделе, должен быть хотя бы один объект.

Если существует несколько разделов с таким компонентом, вам будет предложен выбор раздела.

Внимание! Если во время предпросмотра формы альтернативной формы добавления или изменения вы сохраните данные объекта, то это подействует на систему — в случае добавления, будет добавлен новый объект, в случае изменения — объект будет изменен.

Часть 14. Использование кодировки UTF-8

NetCat, начиная с версии 4.1, может работать как с однобайтной кодировкой (windows-1251), так и с многобайтной (utf-8).
Параметр, который определяет режим работы, задается в файле vars.inc.php и называется $NC_UNICODE. Если он равен «1», то NetCat работает в режиме «utf-8», а если он равен «0», то NetCat считает, что сайт использует однобайтовую кодировку. Так же в файле vars.inc.php нужно задать кодировку страниц и кодировку соединения с базой через параметры $NC_CHARSET и $MYSQL_CHARSET соответственно.

Таким образом, при использовании utf-8 параметры должны быть такими:
$MYSQL_CHARSET = "utf8";
$NC_UNICODE = 1;
$NC_CHARSET = "utf-8";

А при использовании кодировки windows-1251:
$MYSQL_CHARSET = "cp1251";
$NC_UNICODE = 0;
$NC_CHARSET = "windows-1251";

Чтобы перейти от кодировки windows-1251 на utf-8, нужно:

- сконвертировать базу данных в нужную кодировку;
- изменить вышеперечисленные параметры.

Файлы конвертировать не надо.

Использования строковых функций и регулярных выражений

Стандартные php-функции некорректно работают с UTF-8. Например, функция подсчета длины строки (**strlen**("НетКэт") вернет 12, если «НетКэт» будет в кодировке UTF-8.
Чтобы этого избежать, можно использовать расширение mbstring или API NetCat'a.
API NetCat'a для работы со строками содержит следующие функции:

- nc_strlen — аналог strlen
- nc_substr — аналог substr
- nc_strpos — аналог strpos
- nc_strrpos — аналог strrpos
- nc_preg_split — аналог preg_split

- nc_preg_match — аналог preg_match
- nc_preg_match_all — аналог preg_match_all
- nc_preg_replace — аналог preg_replace
- nc_preg_replace_callback — аналог preg_replace_callback
- nc_preg_grep — аналог preg_grep.

Устранение проблем

Если в процессе работы с системой у Вас возникли сложности, которые Вы не смогли разрешить при помощи данного Руководства или Руководства пользователя, возможно, Вы найдете ответ на свой вопрос на сайте netcat.ru в разделе «Поддержка». Там же Вы можете задать вопрос другим разработчикам или производителю системы.

Также Вы можете обратиться к разработчику системы напрямую по телефону горячей линии, указанному на сайте NetCat.ru. Эта возможность доступна зарегистрированным пользователям всех версий системы.

Примечания

Крайне не рекомендуется в названиях своих переменных, функций, классов, названиях дополнительных полей системных таблиц использовать префикс nc или **netcat**, это может привести к проблемам при очередном обновлении системы.

Приложение 1. Описание базы данных

В этом разделе представлена схема базы данных в упрощенном виде (без некоторых второстепенных таблиц). Для каждой таблицы указываются: ее название, описание назначения, а также таблицы, связанные с ней по типу связи «много:1» (например, таблица Subdivision связана с таблицей Catalogue, т.е. для одного сайта (экземпляра сущности Catalogue) может быть определено несколько разделов (экземпляров сущности Subdivision)).

Для реализации связи «много:много» введены т.н. таблицы-связки. Так, таблица Sub_Class является связкой между таблицами Subdivision и Class.

Подробную структуру в синтаксисе SQL Вы можете получить при помощи SQL-консоли, которая входит в поставку системы.

Catalogue
Таблица для хранения настроек сайтов. Структура таблицы расширяется при помощи интерфейса управления системными таблицами. Связанные таблицы: Template, системные списки (системные классификаторы).

Class
Таблица для хранения экземпляров сущностей «Компонент». Поля компонентов хранятся в таблице Field. Связанные таблицы: Module, системные списки (системные классификаторы).

Classificator
Таблица «Классификаторы» («Списки»).

Field
Таблица «Поля компонента». Связанные таблицы: Class, списки.

Filetable
Таблица «Файлы». Здесь хранится информация о закачанных через веб-интерфейс файлах. Связанные таблицы: Field, MessageXX (XX – номер компонента).

Message
Таблицы для хранения данных. Названия таблиц имеют формат MessageXX, где XX – номер компонента (Class), который соответствует таблице. Структура таблиц расширяется при помощи интерфейса управления компонентами. Связанные таблицы: Sub_Class, User, классификаторы.

Permission
Таблица «Экземпляр прав». В таблице хранятся экземпляры прав для конкретных пользователей. Связанные таблицы: PermissionGroup, User.

Redirect
Таблица для хранения ссылок функционала «Переадрессация».

Settings
Таблица с некоторыми настройками системы.

Sub_Class
Таблица-связка между разделом (Subdivision) и компонентов (Class). Используется для составления соответствия между компонентами и разделами (связь между ними «много:много»). Структура таблицы расширяется при помощи интерфейса управления системными таблицами. Связанные таблицы: Subdivision, Class.

Subdivision
Таблица разделов. В ней хранится структура разделов сайтов. Связанные таблицы: Catalogue, Template, классификаторы.

Template
Макеты дизайна. Структура расширяется при помощи интерфейса управления системными таблицами.

User
Пользователи. Структура расширяется при помощи интерфейса управления системными таблицами.

User_Group
Таблица-связка между пользователями и группами. Связанные таблицы: User, PermissionGroup

Приложение 2. Список используемых переменных

В этом приложении приведено подробное описание всех переменных и функций.

Переменные, макропеременные и массивы системы

%Footer

Доступность: хедер и футер макета дизайна

Макропеременная содержит текст футера родительского макета. Если текущий макет -корневой, значение макропеременной будет пустым. Используется в случаях, когда необходимо вывести футер родительского раздела, добавив что-либо до или после него.

Пример использования

Необходимо модифицировать родительский макет, введя еще одну колонку таблицы, выводимую справа от содержательной части страницы и содержащую новости. Текст футера макета:

```
<td valign='top'>".s_list_class(1,2,"")."</td>
%Footer
```

%Header

Доступность: хедер и футер макета дизайна

Макропеременная содержит текст хедера родительского макета. Назначение аналогично макропемеренной %Footer.

%АтрибутСайтаИлиРаздела

Доступность: шаблоны вывода навигации макета дизайна

Используется для генерации HTML-текст навигационных элементов по сайту.Так, элемент навигации «Путь до текущей страницы» (s_browse_path($browse_path)) может иметь такие настройки:

```
$browse_path[prefix] = "";
$browse_path[active] = "%NAME";
$browse_path[active_link] = "%NAME";
$browse_path[unactive] = "<a href=%URL>%NAME</a>";
$browse_path[divider] = " / ";
$browse_path[suffix] = "";
```

Макропеременная %URL обозначает ссылку на сайт/раздел/компонент раздела, который соответствует названию. Другие доступные значения:

%NAME – название
%PARENT_SUB – номер родительского раздела (только для разделов)
%KEYWORD – ключевое слово раздела (только для разделов)
%SUB – номер раздела (только для разделов)
%COUNTER – номер выводимого элемента в списке (начиная с нуля)

Пример использования

Учитывая, что все шаблоны навигации активизируются при помощи функции eval(), при необходимости Вы можете встраивать в эти шаблоны различные функции. Например, шаблон навигации для отображения разделов и подразделов будет выглядеть следующим образом:

Шаблон навигации второго уровня:

```
$browse_sub[1][prefix] = "<font size=-2>";
$browse_ sub[1][active] = "<li><b><a href=%URL>%NAME</a></b>";
$browse_ sub[1][active_link] = "<li><b>%NAME</b>";
$browse_ sub[1][unactive] = "<li><a href=%URL>%NAME</a>";
$browse_ sub[1][divider] = "";
$browse_ sub[1][suffix] = "</font>";
```

Шаблон навигации первого уровня:

```
$isub1 = "\".s_browse_sub(\$data[\$i][Subdivision_ID],\
$browse_sub[1]).\"";
$browse_sub[0][prefix] = "\";global \$browse_sub;\$result.=\"";
$browse_ sub[0][active] = "<li><b><a href=%URL>%NAME</a></b>".
$isub1;
$browse_ sub[0][active_link] = "<li><b>%NAME</b>".$isub1;
$browse_ sub[0][unactive] = "<li><a href=%URL>%NAME</a>".$isub1;
$browse_ sub[0][divider] = "";
$browse_ sub[0][suffix] = "</font>";
```

Таким образом, после каждого элемента меню (раздела) первого уровня вызывается функция отображения его подразделов.

%ПолеМакета

Доступность: хедер и футер макета дизайна

Если таблица Template (в ней хранятся тексты макетов дизайна) содержит дополнительные поля, в хедере и футере макета доступны переменные %ПолеМакета, где ПолеМакета – название поля в таблице MySQL. Управление дополнительными полями (добавление, удаление, изменение свойств) производится на странице «Системные таблицы» системы администрирования.

Пример использования

Необходимо вывести в хедере содержимое поля «Таблица CSS» (в базе данных поле имеет название CSS). Фрагмент текста хедера:

```
...
<style><!-- %CSS_Table --></style>
...
```

$admin_mode

Доступность: макеты и компоненты

Логическая переменная. Истина, если страница выводится в режиме администрирования, и наоборот.

Пример использования

В режиме администрирования для каждого объекта на странице (например, в гостевой книге) необходимо показать IP-адрес добавившего объект. Фрагмент шаблона вывода объекта в списке компонента:

```
$f_Name ".opt($admin_mode, "(IP: $f_IP)")." пишет: $f_Text
```

$begRow

Доступность: шаблоны вывода компонента

Порядковый номер объекта, с которого начинается вывод списка объектов на текущей странице. Нумерация начинается с первого объекта на первой странице. Таким образом, если объекты какого-либо компонента выводятся по 15 штук на странице, на второй странице листинга значение $begRow будет равно 16.

Пример использования

Необходимо в суффиксе вывода списка новостей отобразить номера объектов, выводимых на странице. Фрагмент суффикса вывода списка объектов компонента:

На странице показаны новости $begRow - $endRow

$catalogue

Доступность: везде

Номер текущего сайта. Переменной присваивается значение поля Catalogue_ID таблицы Catalogue, соответствующее текущему сайту.

Пример использования

В макете дизайна, который используется сразу на нескольких сайтах, необходимо ставить ссылку на основной сайт, кроме, разумеется, страниц самого основного сайта. Фрагмент хедера макета:

```
".opt($catalogue!=1, "Посетите наш <a
href=http://www.site.ru>основной сайт</a>!")."
```

$cc

Доступность: везде

Номер (ID) текущего компонента раздела. Переменной присваивается значение поля Sub_Class_ID таблицы Sub_Class, соответствующей текущему компоненту раздела.

Пример использования

На странице необходимо вывести ссылку на ту же страницу в редиме администрирования. Фрагмент хедера/футера макета или суффикса/префикса компонента:

```
<a href=/netcat/?action=index&sub=$sub&cc=$cc>Редактировать эту
страницу</a> (только для модераторов!)
```

$cc_array[]

Доступность: везде

Массив индексов (номеров) компонентов текущего раздела, отсортированных по приоритету. Нулевой элемент ($cc_array[0]) – компонент с наименьшим приоритетом.

Пример использования

Предположим, что для текущего раздела компоненты выводятся не списком на одной странице, а каждый в отдельной «закладке» (см. настройки раздела). Один из компонентов раздела – основной, например, «оглавление раздела», в тексте которого нужно вывести ссылки на все компоненты раздела. Фрагмент системных настроек компонента:

```
$links="";
foreach($some_cc=$cc_array)
{
   if($cc != $some_cc)
      $links .= listQuery("select Sub_Class_Name, EnglishName from
Sub_Class where Sub_Class_ID=$some_cc", "<a href=$subLink\
$data[EnglishName].html>\$data[Sub_Class_Name]</a> ");
}
```

Переменная $links будет доступна в любых полях вывода компонента.

$ccLink

Доступность: шаблоны вывода объектов компонента

Переменная содержит путь к текущему компоненту раздела вида «/about/pr/news.html.

Пример использования

Необходимо вывести ссылку на постоянный адрес страницы этого компонента раздела. Фрагмент суффикса/префикса шаблона вывода компонента:

Постоянный адрес этой страницы: $ccLink

$current_catalogue[], $current_sub[], $current_cc[], $current_user[] (хэш-массивы)

Доступность: везде

Содержат значения свойств текущего каталога, раздела, компонента раздела соответственно, и текущего авторизованного пользователя. Индекс массива должен соответствовать запрашиваемому полю таблицы, например, $current_sub[Subdivision_Name].

Массив $current_user[] может быть установлен только для авторизованного пользователя, он доступен только при наличии установленного модуля «Интерфейс пользователя». Без этого модуля Вы можете узнать ID авторизованного пользователя через переменную $AUTH_USER_ID (при ее отсутствии необходимо сделать ее global).

Пример использования

В таблице «Разделы» (Subdivision) при помощи страницы «Системные таблицы» добавлено поле MetaKeywords, куда прописываются ключевые слова(значения мета-тегов Keywords) страницы. Если для какого-то раздела ключевые слоа не прописаны, используются стандартные ключевые слова «рога, копыта, стерилизация». Фрагмент хедера макета дизайна:

```
<meta name=keywords
content='".opt_case($current_sub[MetaKeywords],
$current_sub[MetaKeywords], "рога, копыта, стерилизация")."'>
```

$endRow

Доступность: шаблоны вывода компонента

Порядковый номер объекта, с которого начинается вывод списка объектов на текущей странице. Нумерация начинается с первого объекта на первой странице.

Пример использования: см. $begRow

$nc_prev_object
$nc_next_object

Доступность: шаблон вывода объекта на одной странице

Данные переменные содержат ссылки на «предыдущий» и «следующий» объект относительно текущего. Для определения порядка объектов используется переменная $query_order из системных настройках, а если она не задана — то значение поля «Сортировать объекты по» в настройках компонента в разделе.

Пример использования:

В шаблоне отображения одного объекта на странице:

```
".opt($nc_prev_object, "<a href='$nc_prev_object'>Предыдущая
новость</a>")."
".opt($nc_next_object, "<a href='$nc_next_object'>Следующая
новость</a>")."
```

$f_AdminButtons

Доступность: шаблоны вывода объекта в списке и объекта на одной странице компонента

В режиме администрирования содержит блок статусной информации об объекте и ссылки на действия для данного объекта «изменить», «удалить», «включить/выключить» (только в поле «Объект в списке»). Если в шаблоне вывода объекта в списке или объекта на одной странице эта переменная указана не будет, вы не сможете изменить или удалить объекты этого компонента стандартными средствами системы.

Пример использования

Для каждого объекта компонента в режиме администрирования выводить блок статусной информации и ссылки на действия для объекта. Фрагмент шаблона вывода объекта в списке:

Имя: $f_Name $f_AdminButtons
Фамилия: $f_LastName

$f_AdminCommon

Доступность: шаблоны вывода компонента

В режиме администрирования содержит блок статусной информации о компоненте раздела и ссылку на добавление объекта в данный компонент раздела и удаление всех объектов из этого же компонента (только в поле «Объект в списке»).

Пример использования

Необходимо в режиме администрирования вывести соответствующий блок перед содержимым компонента раздела. Начало префикса списка объектов:

```
$f_AdminCommon <table...>
```

$f_Checked

Доступность: шаблон вывода объекта в списке и шаблон вывода объекта на одной странице

Истина, если объект включен, и наоборот.

В режиме редактирования необходимо ясно обратить внимание модератора на выключенные объекты. Фрагмент шаблона вывода объекта в списке:

```
".opt($f_Checked, "<font color=red size=+2>объект выключен!!
</a>")."
```

$f_Created

Доступность: шаблон вывода объекта в списке и шаблон вывода объекта на одной странице

Дата и время создания (добавления) объекта в формате "гггг-мм-дд чч:мм:сс". Помимо полного формата вывода даты и времени доступны переменные $f_Created_year, $f_Created_month, $f_Created_day, $f_Created_hours, $f_Created_minutes, $f_Created_seconds, содержащие соответственно год, месяц, день, час, минуту и секунду добавления объекта.

Пример использования

В компоненте "Вакансии" необходимо выводить дату и время создания вакансии в удобной формате. Фрагмент шаблоны вывода объекта в списке:

```
Вакансия добавлена $f_Created_day.$f_Created_month.$f_Created_year
в $f_Created_hours:$f_Created_minutes
```

$f_IP

Доступность: шаблон вывода объекта в списке и шаблон вывода объекта на одной странице

IP посетителя, добавившего этот объект.

Пример использования

Вывести IP посетителя, добавившего объект. Фрагмент шаблоны полного вывода объекта:

```
IP: $f_IP
```

$f_LastIP

Доступность: шаблон вывода объекта в списке и шаблон вывода объекта на одной странице

IP посетителя, последним изменившего этот объект.

Пример использования: аналогично $f_IP

$f_LastUpdated

Доступность: шаблон вывода объекта в списке и шаблон вывода объекта на одной странице

Дата и время последнего изменения этого объекта.

Пример использования: аналогично $f_IP

$f_LastUserAgent

Доступность: шаблон вывода объекта в списке и шаблон вывода объекта на одной странице

Содержимое переменной окружения HTTP_USER_AGENT пользователя, последним изменившего этот объект.

Пример использования: аналогично $f_IP

$f_LastUserID

Доступность: шаблон вывода объекта в списке и шаблон вывода объекта на одной странице

Номер (ID) пользователя, последним изменившего этот объект.

Пример использования: аналогично $f_IP

$f_RowID

Доступность: шаблон вывода объекта в списке и шаблон вывода объекта на одной странице

Номер (ID) объекта в таблице MySQL.

Пример использования: аналогично $f_IP

$f_RowNum

Доступность: шаблон вывода объекта в списке

Порядковый номер объекта на странице. Нумерация начинается с первого объекта, отображенного на странице.

Пример использования: аналогично $f_IP

$f_title

Доступность: макет дизайна

Заголовок текущей страницы. Выводит название текущего раздела за исключением случая, когда текущая страница представляет собой вывод одного объекта на странице и поле «Заголовок страницы» данного компонента определено.

Пример использования

Вывести заголовок страницы в макете. Фрагмент хедера страницы:

<title>$f_title</title>

$f_UserID

Доступность: шаблон вывода объекта в списке и шаблон вывода объекта на одной странице

Номер (ID) пользователя, добавившего этот объект.

Пример использования: аналогично $f_IP

$f_UserAgent

Доступность: шаблон вывода объекта в списке и шаблон вывода объекта на одной странице

Содержимое переменной окружения HTTP_USER_AGENT пользователя, добавившего этот объект.

Пример использования: аналогично $f_IP

$f_ИмяПоля

Доступность: шаблон вывода объекта в списке и шаблон вывода объекта на одной странице

Выводит значение поля с названием "ИмяПоля" данного объекта. Например, если поле "Фамилия" имеет имя LastName, вывести фамилию для текущего объекта можно переменной $f_LastName.

Для разных типов полей доступны также следующие варианты суффиксов:

- Дата и время: $f_ИмяПоля_year, $f_ИмяПоля_month и т.д. - аналогично переменной $f_Created
- $f_ИМЯПОЛЯ_name, $f_ИМЯПОЛЯ_type, $f_ИМЯПОЛЯ_size
переменные, содержащие соответственно оригинальное имя закачанного файла, его тип (mime type) и размер в байтах. При этом переменная $f_ИМЯПОЛЯ содержит путь к файлу на сервере. Только для полей типа «Файл»
- $f_ИМЯПОЛЯ_id
содержит ID элемента типа «Список». При этому $f_ИМЯПОЛЯ содержит его название.
- $f_ИМЯПОЛЯ_name содержит элемент типа “Список” в действии после добавления

Пример использования: аналогично $f_IP

$fullDateLink

Доступность: шаблон вывода объекта в списке и шаблон вывода объекта на одной странице

Ссылка на страницу с полным выводом объекта в виде «…/2002/02/02/message_2.html» (устанавливается в случае если в компоненте имеется поле типа «Дата и время» с форматом «event», иначе значение переменной идентично значению $fullLink).

Пример использования

В списке новостей около анонса каждой новости выводить ссылку «подробнее» на полный текст страницы. Фрагмент шаблона вывода объекта в списке:

```
$f_Anons // <a href=$fullDateLink>подробнее</a>
```

$fullLink

Доступность: шаблон вывода объекта в списке и шаблон вывода объекта на одной странице

Ссылка на страницу с полным выводом объекта. Ссылка формируется в «человеко-понятном» формате:
/URLРаздела/КлючСловоКомпонентаРаздела_КлючСловоОбъекта.html
или
/URLРаздела/КлючСловоОбъекта.html, например, /about/team/ivanov.html

Пример использования: аналогично $fulldateLink

$editLink
$deleteLink
$dropLink
$checkedLink

Доступность: шаблон вывода объекта в списке и шаблон вывода объекта на одной странице

Ссылки на страницу с редактированием, удалением, удалением без подтверждения, включением-выключением объекта соответственно.

$MODULE_VARS[КлючевоеСловоМодуля][ИмяПеременной]

Массив содержит значения переменных настроек модулей.

Пример использования

В компоненте "результаты поиска" необходимо указать все сайты, индексируемые поисковым роботом NetCat. Фрагмент суффикса списка объектов:

Внимание!

Поиск ведется на сайтах:″. $MODULE_VARS[search][ALLOWED_URLS].″

$nextLink
Доступность: поля шаблона вывода объектов компонента

Переменная содержит ссылку на следующую страницу в листинге компонента (если текущее положение в списке – его конец, то переменная пустая).

Пример использования

После вывода списка объектов необходимо вывести ссылку на следующую страницу. Фрагмент суффикса списка объектов:

```
".opt($nextLink, "<a href=$nextLink>далее</a>")."
```

$parent_sub_tree[]

Доступность: макеты дизайна, компоненты

Массивы свойств раздела различных уровней вложенности в реверсивном порядке (от текущего раздела $parent_sub_tree[0] до свойств головного сайта $parent_sub_tree[$sub_level_count-1]).

Пример использования

Идентификатор текущего раздела - $parent_sub_tree [0][Subdivision_ID]
Идентификатор родительского раздела - $parent_sub_tree [1][Subdivision_ID]

$prevLink

Доступность: поля шаблона вывода объектов компонента

Переменная содержит ссылку на следующую страницу в листинге компонента (если текущее положение в списке – его конец, то переменная пустая).

Пример использования: аналогично $nextLink

$recNum

Доступность: поля шаблона вывода объектов компонента

Максимальное количество объектов для вывода. Переменную можно подать в адресной строке (/news/?recNum=3), в функции s_list_class(), в системных настройках компонента.

Пример использования

На титульной странице сайта необходимо вывести 3 последних новости. Раздел "Новости" имеет ID 1, нужный компонент раздела "Новости"- ID 2. Фрагмент футера или хедера макета титульной страницы:

```
".s_list_class(1, 2, "&recNum=3")."
```

$sub

Доступность: везде

Номер (ID) текущего раздела.

Пример использования: см. $cc

$sub_level_count

Доступность: макеты дизайна

Переменная содержит текущий уровень вложенности навигации.

Пример использования

Уровень вложенности от корня сайта - $sub_level_count

$subHost

Доступность: везде

Переменная содержит текущий хост (домен) вида «www.company.ru».

$subLink

Доступность: везде

Переменная содержит URI текущего раздела вида «/about/news/».

$totRows

Доступность: поля редактирования шаблонов вывода компонента

Содержит общее количество объектов данного компонента раздела.

Пример использования

Необходимо вывести общее количество объектов компонента раздела (например, вакансий). Фрагмент префикса шаблона вывода списка объектов:

Всего вакансий в базе: $totRows.

Приложение 3. Список функции

Функции навигации и листинга

browse_messages($cc_env, $range)

Доступность: шаблоны вывода компонентов

Отображает блок навигации по страницам списка объектов компонента в формате «1 2 3 >>». Массив $cc_env является неизменным параметром данной функции и содержит переменные окружения текущего компонента раздела. Параметр $range определяет количество выводимых страниц. Вместо этой переменной обычно пишется число. Подразумевается, что из множества страниц одновременно будет показываться только список из $range страниц. Например, ваш листинг состоит из 20 страниц. Если $range=10, то, находясь на первой странице, вы будете видеть страницы с 1 по 10, находясь на 15-й странице, вы будете видеть страницы 10-20. Для настройки формата отображения используется массив $browse_msg[], значения которого указываются в настройках макета дизайна. Макропеременная %PAGE обозначает номер страницы, макропеременная %URL – ссылку на соответствующие страницы. Возможно также использование макропеременных %FROM и %TO, соответственно обозначающих номера начального и конечного объекта на странице.

```
$browse_msg[prefix] = ""; // Префикс перед блоком навигации
$browse_msg[suffix] = ""; // Суффикс после блока навигации
$browse_msg[active] = "%PAGE"; // Формат вывода текущего
$browse_msg[unactive] = "<a href=%URL>%PAGE</a>"; // Формат вывода
ссылок
$browse_msg[divider] = " "; // Разделитель между ссылками
```

Пример использования

Необходимо внизу списка объектов выводить постраничную навигацию по 15 ссылок (Страницы 1 2 3 ... 15). Фрагмент суффикса списка объектов компонента:

```
".s_browse_messages($cc_env, 15)."
```

s_browse_catalogue($template)

Доступность: макеты дизайна

Функция выводит список сайтов (блок навигации) в соответствии с шаблоном, описанным в хэш-массиве $template. Массив должен иметь элементы со следующими индексами:

- prefix – выводится перед списком
- suffix – выводится после списка
- active – шаблон вывода активного элемента списка (а данном случае это касается текущего сайта)
- active_link – шаблон вывода активного элемента списка в том случае, если ссылка на этот элемент идентична адресу текущей страницы
- unactive – шаблон вывода неактивного элемента списка (в данном случае это касается всех сайтов, кроме текущего)
- divider – шаблон разделителя между элементами списка
- sortby – признак сортировки элементов

Пример использования

В макете страницы необходимо вывести список всех сайтов в виде нумерованного списка. Текущий сайт ссылкой выделяться не должен. Фрагмент хедера или футера макета:

```
".s_browse_catalogue($cat_template)."
```

Фрагмент шаблона вывода навигации макета:

```
$cat_template[prefix] = "<ul>";
$cat_template[suffix] = "</ul>";
$cat_template[active] = "<li> %NAME</li>";
$cat_template[active_link] = "<li> %NAME</li>";
$cat_template[unactive] = "<li> <a href=%URL>%NAME</a></li>";
$cat_template[divider] = "";
$cat_template[sortby] = "Priority DESC";
```

s_browse_cc($template)

Доступность: макеты дизайна

Выводит список ссылок по компонентам раздела в соответствии с шаблоном $template.

Пример использования: аналогично s_browse_catalogue()

s_browse_sub(int $parent_sub, $template, $ignore_check = 0, $where_cond = '')

Доступность: макеты дизайна

Выводит список подразделов раздела $parent_sub в соответствии с шаблоном $template. С помощью флага $ignore_check можно игнорировать вывод только включённых разделов (если $ignore_check равен 1, то выведутся все разделы). С помощью $where_cond можно дополнить запрос в секции WHERE
Пример использования: аналогично s_browse_catalogue()

s_browse_level(int $level, $template, $ignore_check = 0, $where_cond = '')

Доступность: макеты дизайна

Выводит список разделов уровня $level в соответствии с шаблоном $template. Обратите внимание, что нумерация уровней начинается с нуля, т.е. Для вывода списка разделов верхнего уровня первый параметр функции должна быть равен нулю.

Пример использования: аналогично s_browse_catalogue()

s_browse_path($template)

Доступность: макеты дизайна

Выводит навигацию типа "хлебные крошки" (путь до текущей страницы) в соответствии с шаблоном $template.

Пример использования: аналогично s_browse_catalogue()
s_browse_path_range($from,$to,$template)

Доступность: макеты дизайна

функция аналогична s_browse_path(), но выводит только путь указанного диапазона (минимальное значение $from – (-1), максимальное значение $to - $sub_level_count), в соответствии с шаблоном, описанным в массиве $template. Например: s_browse_path_range(0,$sub_level_count, $template) выведет навигацию от корня сайта, но без текущего раздела.

Функции генерации полей для альтернативных форм добавления и изменения

nc_put_field(*string* ***$field_name,*** *[string* ***$style,*** *[int* ***$classID,*** *[bool* ***$caption****]]]***)**

Функция генерирует поле любого типа.

- $field_name — имя поля в текущем компоненте или компоненте $classID.
- $style (опционально) — параметры отображения поля, например: «size='50' class='my_field'».
- $classID (опционально) — идентификатор компонента, используется например, при выводе формы через функцию s_list_class() его следует указывать, по умолчанию этот параметр можно не задавать.
- $caption (опционально) — выводить или не выводить описание над генерируемым полем.

nc_list_field(*string* ***$field_name,*** *[string* ***$style,*** *[int* ***$classID,*** *[bool* ***$caption,*** *[mixed* ***$selected,*** *[mixed* ***$disabled, [****bool* ***$ignore_check****]]]]]]***)**

Функция генерирует поле типа «Список».

- $field_name — имя поля в текущем компоненте или компоненте $classID.
- $style (опционально) — параметры отображения поля, например: «class='my_field'».
- $classID (опционально) — идентификатор компонента, используется например, при выводе формы через функцию s_list_class() его следует указывать, по умолчанию этот параметр можно не задавать.
- $caption (опционально) — выводить или не выводить описание над генерируемым полем.
- $selected (опционально) — выбранный элемент списка.
- $disabled (опционально) — выключенный элемент списка.
- $ignore_check — игнорировать выборку только включенных

nc_file_field(*string* ***$field_name,*** *[string* ***$style,*** *[int* ***$classID,*** *[bool* ***$caption****]]]***)**

Функция генерирует поле типа «Файл». В большинстве случаев она применяется в альтернативных формах компонента, чтобы вывести информацию о поле типа Файл.

- $field_name — имя поля в текущем компоненте или компоненте $classID.
- $style (опционально) — параметры отображения поля, например: «size='50'».
- $classID (опционально) — идентификатор компонента, используется например, при выводе формы через функцию s_list_class() его следует указывать, по умолчанию этот параметр можно не задавать.
- $caption (опционально) — выводить или не выводить описание над генерируемым полем.

Пример использования

В альтернативной форме добавления объекта вывести поле для закачки файла, соответствующее полю компонента Photo. Фрагмент альтернативной формы добавления:

```
".nc_file_field("Photo", "size=100 style='color:red;'")."
```

nc_bool_field(*string* ***$field_name,*** *[string* ***$style,*** *[int* ***$classID,*** *[bool* ***$caption****]]]***)**

Функция генерирует поле типа «Логическая переменная».

- $field_name — имя поля в текущем компоненте или компоненте $classID.
- $style (опционально) — параметры отображения поля, например: «size='50'».
- $classID (опционально) — идентификатор компонента, используется например, при выводе формы через функцию s_list_class() его следует указывать, по умолчанию этот параметр можно не задавать.
- $caption (опционально) — выводить или не выводить описание над генерируемым полем.

nc_date_field(*string* ***$field_name,*** *[string* ***$style,*** *[int* ***$classID,*** *[bool* ***$caption,*** *[string* ***$dateDiv,*** *[string* ***$timeDiv****]]]]]***)**

Функция генерирует поле типа «Дата и время».

● $field_name — имя поля в текущем компоненте или компоненте $classID.
● $style (опционально) — параметры отображения поля, например: «size='50'».
● $classID (опционально) — идентификатор компонента, используется например, при выводе формы через функцию s_list_class() его следует указывать, по умолчанию этот параметр можно не задавать.
● $caption (опционально) — выводить или не выводить описание над генерируемым полем.
● $dateDiv (опционально) — разделитель полей для даты, по умолчанию "-".
● $timeDiv (опционально) - разделитель полей для времени, по умолчанию ":".

nc_text_field(*string* ***$field_name,*** *[string* ***$style,*** *[int* ***$classID,*** *[bool* ***$caption,*** *[bool* ***$bbcode****]]]]***)**

Функция генерирует поле типа «Текстовый блок».

● $field_name — имя поля в текущем компоненте или компоненте $classID.
● $style (опционально) — параметры отображения поля, например: «size='50'».
● $classID (опционально) — идентификатор компонента, используется например, при выводе формы через функцию s_list_class() его следует указывать, по умолчанию этот параметр можно не задавать.
● $caption (опционально) — выводить или не выводить описание над генерируемым полем.
● $bbcode (опционально) - выводить панельку с BB-кодами (для панельки нужны стили CSS в макете дизайна!), по умолчанию — отключено.

nc_string_field(*string* ***$field_name,*** *[string* ***$style,*** *[int* ***$classID,*** *[bool* ***$caption****]]]***)**

Функция генерирует поле типа «Строка».

● $field_name — имя поля в текущем компоненте или компоненте $classID.
● $style (опционально) — параметры отображения поля, например: «size='50'».

● $classID (опционально) — идентификатор компонента, используется например, при выводе формы через функцию s_list_class() его следует указывать, по умолчанию этот параметр можно не задавать.
● $caption (опционально) — выводить или не выводить описание над генерируемым полем.

nc_int_field(*string* ***$field_name,*** *[string* ***$style,*** *[int* ***$classID,*** *[bool* ***$caption****]]]***)**

Функция генерирует поле типа «Целое число».

● $field_name — имя поля в текущем компоненте или компоненте $classID.
● $style (опционально) — параметры отображения поля, например: «size='50'».
● $classID (опционально) — идентификатор компонента, используется например, при выводе формы через функцию s_list_class() его следует указывать, по умолчанию этот параметр можно не задавать.
● $caption (опционально) — выводить или не выводить описание над генерируемым полем.

nc_float_field(*string* ***$field_name,*** *[string* ***$style,*** *[int* ***$classID,*** *[bool* ***$caption****]]]***)**

Функция генерирует поле типа «Число с плавающей запятой».

● $field_name — имя поля в текущем компоненте или компоненте $classID.
● $style (опционально) — параметры отображения поля, например: «size='50'».
● $classID (опционально) — идентификатор компонента, используется например, при выводе формы через функцию s_list_class() его следует указывать, по умолчанию этот параметр можно не задавать.
● $caption (опционально) — выводить или не выводить описание над генерируемым полем.

nc_multilist_field(*string* **$field_name,** *[string* **$style,** *[string* **$type,** *[int* **$classID,** *[bool* **$caption,** *[string* **$selected,** *[bool* **$disabled,** *[bool* **$getData***]]]]]]***)**

Функция генерирует поле типа «Множественный выбор»

Аргументы:

- $field_name - Имя поля
- $style - Дополнительные атрибуты
- $type - Тип: "select", "select:N", "checkbox", по умолчанию - "select:3"
- $classID - Идентификатор компонента
- $caption - Выводить описание поля или нет
- $selected - Выбранные элементы. Передаются через строку, где id выбранных элементов разделены между собой символами '[пробел]', '.', ',', ';'
- $disabled -Недоступные элементы. Формат аргумента аналогичен предыдущему
- $getData - Принудительно выстакивать из базы

nc_list_select ($classificator_name, $field_name = NULL, $current_value = NULL, $sort_type = NULL, $sort_direction = NULL, $template_prefix = NULL, $template_object = NULL, $template_suffix = NULL, $template_any = NULL, $ignore_check = false)

Доступность: везде

Данная функция позволяет генерировать HTML списки из Списков NetCat. В большинстве случаев она применяется в альтернативных формах компонента, чтобы вывести информацию о поле типа Список.

- $classificator_name – имя списка, например, Gallery, обязательный параметр;
- $field_name – название поля в компоненте (без префикса f_, например, Field);
- $current_value – выбранный элемент списка (например, $f_Field_id, если мы используем фунцию в альтернативной форме добавления/изменения);
- $sort_type – поле сортировки, необязательный параметр (не указан – ID, 1 – имя, 2 - приоритет);
- $sort_direction - порядок сортировки, необязательный параметр (не указан – восходящий, 1 - нисходящий);
- $template_prefix – темплейт префикса списка, необязательный параметр (не указан - "<select name='f_\$field_name'>\r\n"), ;
- $template_object – темплейт элемента списка, необязательный параметр (не указан - "<option value='${value_id}' ${value_selected}>$ {value_name}</option>");

- $template_suffix – темплейт суффикса списка, необазательный параметр (не указан - "</select>\r\n");
- $template_any – темпелейт для первого нулевого элемента списка, если поле может быть пустым (не указан - <option value="">--выбрать--</option>);
- $ignore_check — игнорировать выборку только включенных.

Не забывайте экранировать кавычки!

Поле $template_prefix имеет «константу». Переменная, а точнее запись вида «\$field_name», автоматически заменится на указанный в вызове функции параметр $field_name.

Поле $template_object имеет 3 «константы»: «\$value_id», «\$value_selected» и «\$value_name», которые отвечают за подстановку ID, выбранной записи и название элемента.

Пример использования

Необходимо сгенерировать в альтернативной форме изменения объекта HTML-код для отображения поля GalleryName из компонента, которое использует список Gallery. Сортировка по ID нисходящая. Имеется 3 темплейта: префикс, элемент списка и суффикс. Темпелейт для первого нулевого элемента списка (если поле является необязательным для заполнения) будет использован по умолчанию из функции.

```
".nc_list_select("Gallery", "GalleryName", $f_GalleryName_id,"",1,
"<select name='f_\$field_name'>", "<OPTION value='\$value_id' \
$value_selected>\$value_name</OPTION>")."
```

Функции для работы с объектами

nc_message_link (int $message_id, int $class_id, $action='')

Доступность: везде

Функция позволяет получить относительный путь к объекту (без домена) по номеру (ID) этого объекта и номеру (ID) компонента, которому он принадлежит.

1. $message_id – номер объекта;
2. $class_id – номер компонента.
3. $action – действие с объектом

По умолчнию действие — просмотр, возможны варианты:
edit — редактирование
delete — удаление
drop — удаление без подтверждения
checked — смена статуса
Будьте внимательны, не путайте номера компонента и номер компонента раздела. Номер компонента Вы можете узнать, например, в разделе «Список компонентов» системы администрирования.

Пример использования

Необходимо вывести ссылку на объект номер 52 в компоненте номер 2. Фрагмент шаблона:

```
".nc_message_link(52,2)."
```

parentofmessage($message, $classID)

Доступность: везде

Возвращает номер объекта – родителя «ветки» объектов (может использоваться в иерархическом форуме), в которой, в частности, содержится объект с номером $message. Параметр $classID определяет номер компонента.

Пример использования

В простом иерархическом форуме необходимо вывести ссылку на уровень "вверх". Фрагмент шаблона вывода объекта:

```
".opt($parent=parentofmessage($f_RowID, $classID), "")."
".opt($parent,          "<a          href=".nc_message_link($parent,
$classID).">наверх</a>")."
```

s_list_class(int $sub, int $cc, char $params, bool $show_in_admin_mode = FALSE)

Доступность: везде

Функция выводит "верхние" объекты из раздела $sub компонента раздела $cc с параметрами $params. Параметры подаются в URL-формате: &param1=12&param2=46. Последний параметр функции необязателен: если он имеет значение TRUE (истина), этот блок будет выведен и в режиме администрирования, иначе только в обычном режиме работы сайта.

Третий параметр функции - $params – может содержать произвольный состав параметров. Все они могут быть обработаны в тексте шаблона вывода объекта.

Пример использования

В макете дизайна титульной страницы необходимо вывести список последних трех новостей, но без листинга и в укороченном формате. Номер раздела новостей 1, номер компонента раздела новостей – 2. Фрагмент хедера/футера макета:

```
".s_list_class(1, 2, "recNum=3&isTitle=1")."
```

Необходимо обратить внимание, что название isTitle выбрано произвольно. Этот параметр будет "виден" в тексте шаблона вывода. Например, он может быть использован для того чтобы не выводить листинг по страницам компонента. Для этого обратимся к нему в суффиксе списка вывода объектов компонента "Новости":

```
".opt(!$isTitle, browse_messages($cc_env, 10))."
```

А в списке объектов компонента помимо "полного" (обычного) вида списка новостей укажем укороченный:

```
".opt_case($isTitle, "<a href=$fullLink>$f_Title</a>
($f_Created)<br>",
"<a href=$fullLink>$f_Title</a> ($f_Created)<br>$f_Anons<p>")."
```

Функции для работы с группами пользователей

nc_usergroup_create (string $name)

Функция создает группу пользователей.
Аргументы:

- $name — имя группы

nc_usergroup_rename (int $PermissionGroupID, string $name)

Функция переименовывает группу пользователей.
Аргументы:

- $PermissionGroupID — идентификатор группы
- $name — имя группы

nc_usergroup_delete (mixed $PermissionGroupID **)**

Функция удаляет группу пользователей. Группу нельзя удалить, если есть хотя бы один пользователь, который состоит только в этой группе. Функция возвращает массив с номерами удаленных групп.
Аргументы:

- $PermissionGroupID — идентификатор группы или массив с идентификаторами групп

nc_usergroup_add_to_group (int $UserID**, int** $PermissionGroupID **)**

Функция добавляет пользователя в группу.
Аргументы:

- $UserID — идентификатор пользователя
- $PermissionGroupID — идентификатор группы

nc_usergroup_remove_from_group (int $UserID**, int** $PermissionGroupID **)**

Функция исключает пользователя из группы. (Пользователь должен состоять хотя бы в одной группе.)
Аргументы:

- $UserID — идентификатор пользователя
- $PermissionGroupID — идентификатор группы

nc_usergroup_get_users_from_group ($PermissionGroupID , $output_type = 0)

Функция возвращает всех пользователей, находящихся в группе PermissionGroupID.
Аргументы:

- $PermissionGroupID — идентификатор группы
- $output_type — тип выходного массива

Второй параметр позволяет определять структуру выходного массива.
Если output_type равен 0, то на выходе одномерный числовой массив, значения элементов которых соответствует идентификатором пользователей, входящих в эту группу.
Если output_type равен 1 то на выходе одномерный массив, значения элементов которых соответствует логинам пользователей, а индекс — номерам пользоватлей.

nc_usergroup_get_group_by_user ($UserID , $output_type = 0)

Функция возвращает все группы, в которых состоит пользователь.
Аргументы:

- $UserID — идентификатор группы
- $output_type — тип выходного массива

Второй параметр позволяет определять структуру выходного массива.
Если output_type равен 0, то на выходе одномерный числовой массив, значения элементов которых соответствует идентификатором групп.
Если output_type равен 1 то на выходе одномерный массив, значения элементов которых соответствует названиям групп, а индекс — номерам групп.

Другие функции

is_even(int $param)

Доступность: везде

Проверяет параметр на четность.

Пример использования

Необходимо чередовать цвета строк в таблице при выводе объектов компонента. Фрагмент шаблона вывода объекта в списке:

```
<td ".opt_case(is_even($some_counter), "bgcolor=white",
"bgcolor=silver")."">$f_Value</td>
".opt($some_counter++, "")."
```

listQuery(char $sql_query, char $output_template = NULL, $divider = '')

Доступность: везде

Функция производит запрос $sql_query к базе данных, форматирует в соответствии с шаблоном $output_template и выводит результаты запроса. Чаще всего функция используется за запросов SELECT. В шаблоне $output_template могут использоваться обращения к массиву $data[] с маскировочным слешем перед знаком $: \$data[Name]. В качестве индексов массива используются названия столбцов таблиц, из которых происходит выборка. Между строками результата вставляется разделитель $divider.

Примеры использования

Вывод списка всех включенных пользователей системы в порядке очередности регистрации в тексте макета или компонента:

```
".listQuery("select Login from User where Checked=1 order by
Created", "<li> \$data[Login]</li>\n")."
```

Вывод в выпадающем списке всех значений списка Money:

```
<select name='MoneyList'>
".listQuery("SELECT Money_ID, Money_Name FROM
Classificator_Money","<option value=\$data[Money_ID]>
\$data[Money_Name]")."
</select>
```

Будьте осторожны, используя эту функцию: при помощи нее Вы получаете прямой доступ к базе данных.

nc_bbcode($text, $cut_link="", $cut_full="", $codes="")

Доступность: везде

Обрабатывает текст с BB-кодами. Подробнее см. главу «Использование BB-кодов».

nc_bbcode_bar($winID, $formID, $textareaID, $help="", $codes="", $prefix="", $suffix="")

Доступность: везде

Отображает панельку вставки BB-кодов. Подробнее см. главу «Использование BB-кодов».

nc_array_to_string(**$arr,** *$template*)

Доступность: везде

Переводит массив в строку по шаблону.

- $arr - непосредственно сам массив
- $template - массив с шаблонами, должен иметь следующие ключи:
    1. prefix

2. element
3. divider
4. suffix

В элементе с ключом elemnet можно использовать макропеременные:

- %ELEMENT - текущий элемент массива $arr
- %KEY - его ключ
- %I - номер по порядку (отсчет начинается с 1)

Пример использования

Пример 1.

Допустим, есть компонент с полем pole типа множественный выбор. Требуется вывести в «списке объектов» элементы в виде списка

В префиксе запишем

```
";
$templ['prefix'] = '<ui>';
$templ['element'] = '<li>%ELEMENT</li>';
$templ['suffix'] = '</ui>';
$result .= "
продолжение префикса
```

Список объектов:

```
<b>Объект $f_RowID</b>
<br>
".nc_array_to_string($f_pole, $templ)."
<br>
```

Результат:

Объект 3
Ярославль
Ялта
Якутск
Объект 2
Павлодар
Павловск
Хабаровск
Объект 1
Ханты-Мансийск

Москва

HTML-код:

```
<b>Объект 3</b><br>
<ui><li>Ярославль</li><li>Ялта</li><li>Якутск</li></ui>
<br>

<b>Объект 2</b><br>
<ui><li>Павлодар</li><li>Павловск</li><li>Хабаровск</li></ui>
<br>

<b>Объект 1</b><br>
<ui><li>Ханты-Мансийск</li><li> Москва</li></ui>
<br>
```

Пример 2.

В системной таблице «Разделы» добавлено поле pole типа «Множественный выбор»

В макете дизайна:

```
".nc_array_to_string($f_pole,
array( 'prefix' => 'List:<br>',
'element' => '%I. %ELEMENT',
'divider' => '<br>',
'suffix' => 'end' )
)."
```

Так же возможен варинт, когда массив с шаблоном задается в «Шаблоны вывода навигации»

nc_mail2queue($recipient, $from, $subject, $message)

Доступность: везде

Функция является частью функционала по рассылке писем. Указанное письмо кладется в таблицу БД mail_queue, откуда в дальнейшем будет разослано скриптом /netcat/admin/mailer.php.

- $recipient содержит адрес получателя;

- $from содержит координаты отправителя;
- $subject – тема письма;
- $message – сообщение.

При желании Вы можете отсылать даже письма с файлами – для этого при помощи каких-либо средств необходимо сгенерировать правильный $message. Однако учтите, что письмо кладется в БД ровно столько раз, сколько подписчиков его должны получить, и при большом вложенном файле размер БД может значительно вырасти.

Файл /netcat/admin/mailer.php, отвечает за порционную рассылку писем. Количество писем задается параметром number, если не указано – высылается 20 писем из очереди. Этот скрипт должен быть добавлен в CRON Вашего сервера (каждую минуту, /netcat/admin/mailer.php?number=20). При наличии писем в очереди они будут отсылаться порциями. При отсутствии писем скрипт ничего делать не будет.

Пример использования

Необходимо отправлять на почту администратору сайта все новые сообщения из гостевой книги. Фрагмент шаблона действия после добавления компонента:

```
".nc_mail2queue("admin@mysite.ru", "noreply@mysite.ru", "Новое
сообщение в гостевуху", $f_Message)."
```

nc_transliterate($text)

Доступность: везде

Переводит русский текст в "транслит", а именно заменяет кириллические буквы или сочетания букв по массиву соответствий.

Пример использования

Нужно перевести текст "Шла Саша по шоссе и сосала сушку":

```
<?php
echo nc_transliterate("Шла Саша по шоссе и сосала сушку");
?>
```

Результат

```
Shla Sasha po shosse i sosala sushku
```

nc_get_visual_settings(int $cc)

Функция возвращает текущие визуальные настройки компонента в разделе по его идентификатору. Возвращаемый результат — массив данных.

- $field_name — идентификатор компонента в разделе.

nc_file_path ($class_id, $message_id, $field_name, $file_name_prefix="")

Функция позволяет получить путь к файлу, указанному в определенном поле, по номеру (ID) этого объекта и номеру (ID) шаблона, которому он принадлежит.

- $class_id – номер шаблона (для системных таблиц – название таблицы, допустимые значения – “Catalogue”, “Subdivision”, “User”, “Template”);
- $message_id – номер объекта;
- $field_name – название поля или его идентификатор в шаблоне;
- $file_name_prefix (опционально) – укажите “h_”, если хотите получить ссылку для скачивания файла под оригинальным именем (подробнее см. в разделе «Файловая система» настоящего руководства).

Будьте внимательны, не путайте номера шаблона и номер шаблона в разделе. Номер шаблона вы можете узнать, например, в разделе «Список шаблонов» системы администрирования.

Если в указанном поле файл не был закачан, функция возвращает NULL.
При неправильно указанном имени или идентификаторе поля функция сообщит об ошибке и вернет NULL.
Функция работает как с файлами, закачанными в любой версии NetCat (2.2, 2.3, 2.4).

Пример использования

- Путь к файлу на диске в поле “Pic” объекта номер 100 в шаблоне 2:

```
$DOCUMENT_ROOT".nc_file_path(2, 100, 'Pic')."
```

●Если поле “Pic” в шаблоне номер 2 имеет идентификатор 84, то следующий пример аналогичен предыдущему:

```
$DOCUMENT_ROOT".nc_file_path(2, 100, 84)."
```

●Ссылка для скачивания файла, указанного в поле “ForumAvatar” у пользователя с идентификатором 5:

```
<img src=".nc_file_path('User', 5, 'ForumAvatar', 'h_')." />
```

opt($flag, $string)

Доступность: везде

Функция выводит строку $string в случае если $flag не ложь, не имеет пустое значение или не ноль.

Пример использования

В шаблоне вывода объекта в списке выводить фамилию с подписью «Фамилия» только в том случае, если это поле заполнено. Фрагмент шаблона вывода объекта:

```
".opt($f_LastName, "Фамилия: $f_LastName<br>")."
```

opt_case($flag, $string1, $string2)

Доступность: везде

Выводит строку $string1 в случае если $flag не ложь, не имеет пустое значение или не ноль; в противном случае выводит строку $string2.

Пример использования: по аналогии с opt()

Приложение 4. Классы системы

CMIMEMail

Доступность: везде

Класс является частью функционала по рассылке писем. Позволяет отсылать одиночные письма, в том числе и с вложениями. Формирует заголовки писем, позволяя наиболее правильно соответствовать совместимости с разного рода почтовыми сервисами и программами.

Инициализация

Для работы с классом необходимо создать объект этого класса:

$mailer = new CMIMEMail();

Кодировка письма

Для смены кодировки письма требуется вызвать метод класса setCharset(), указав в параметре требуемую кодировку. По умолчанию используется windows-1251 в случае использовании русского языка и ISO-8859-1 для английского.

```
$mailer->setCharset('koi8r');
```

Типы писем, создание тела письма

Класс позволяет отправлять письма в виде простого текста, html-письма и письма с вложениями (с прикрепленными файлами).
Самый просто случай — это письма с простым текстом без вложения. В этом случае необходимо вызвать метод класса mailbody, указав в параметре текст письма.

```
$mailer->mailbody("Я письмо!");
```

или так :

```
$text = "Простой текст простого письма";
$mailer->mailbody($text);
```

В случае html-письма без вложения письмо отправляется в двух видах: простом тексте и html-виде (тип содержания в этом случае multipart/alternative). Если

почтовый клиент не поддерживает html (или показ сообщения html был отключен), то будет показываться простой тескт, иначе — текст в виде html. Простой текст задается первым параметром метода mailbody, html-текст — вторым.

```
$text = "<b>Привет!</b>";
$mailer->mailbody( strip_tags($text), $text);
```

В этом случае, если пользовательский почтовый клиент может показывать html-письма выведется «**Привет!**», а если нет — то просто текст «Привет!». Возможен такой варинт:

```
$mailer->mailbody("К сожалению, Ваш почтовый клиент не может
показать это письмо, так как оно в виде html", $text);
```

Прикрепление файлов

Для создания вложения нужно вызвать метод attachFile, передав ему путь к файлу, имя файла и его тип.

```
$mailer->attachFile("../attach.doc", "имя_файл.txt",
"application/octet-stream");
```

Чтобы прикрепить несколько файлов, этот метод нужно вызвать нужное количество раз:

```
$mailer->attachFile("../1h.doc", "1.txt", "application/octet-
stream");
$mailer->attachFile("../2.doc", "2.txt", "application/octet-
stream");
$mailer->attachFile("../2.doc", "3.txt", "application/octet-
stream");
```

Отправка письма

Для отправки письма нужно вызвать метод send, передав в параметрах адрес назначения, с какого адреса пришло письмо, адрес для ответа, тему письма и имя отправителя.

```
$to = "кому@адрес.ru";
$from = "от_кого@адрес.ru";
$reply = "ответ_прислать_сюда@адрес.ru";
$from_name = "Имя отправителя";
$subject = "Тема письма";

$mailer->send($to, $from, $reply, $subject, $from_name);
```

Итоговый пример использования

```
$to = "кому@адрес.ru";
$from = "от_кого@адрес.ru";
$reply = "ответ_прислать_сюда@адрес.ru";
$from_name = "Имя отправителя";
$subject = "Тема письма";

$text = "<b>Привет!</b>";

$mailer = new CMIMEMail();
$mailer->mailbody( strip_tags($text), $text);
$mailer->attachFile("../attach.doc", "имя_файл.txt",
"application/octet-stream");

$mailer->send($to, $from, $reply, $subject, $from_name);
```

Permission

Доступность: везде

Класс предназначен для проверки прав пользователя.

Константы
Существует несколько констант, которые непосредственно не относятся к классу, но тесно с ним связаны. Константы обозначают возможности. В таблице приведён список этих констант.

Константа	Значение	Право
MASK_READ	1	Просмотр
MASK_ADD	2	Добавление объектов
MASK_EDIT	4	Изменение своих объектов
MASK_SUBSCRIBE	8	Подписка
MASK_MODERATE	16	Модерирование
MASK_ADMIN	32	Администрирование
MASK_COMMENT	64	Комментирование
MASK_CHECKED	128	Включение/выключение своих объектов
MASK_DELETE	256	Удаление своих объектов

Константы можно группировать через побитовые операции, например & (битовое «И») или | (битовое «ИЛИ»).
Например:
(MASK_ADD | MASK_EDIT) обозначает возможность добавления объектов или редактирования своих.
(MASK_EDIT & MASK_DELETE) обозначает возможность редактирования и удаления своих объектов.
Примеры непосредственного использования констант будут приведены ниже в описаниях методов.

Объекты класса
В глобальной области видимости доступен объект этого класса с именем **$perm**, при условии, что пользователь авторизирован. Если пользователь неавторизирован, то объект не существует. Объект содержит права текущего авторизированного пользователя.
Перед использованием этого объекта нужно убедиться, что он существует.
Проверить это можно, можно с помощью переменной $AUTH_USER_ID (который содержит номер текущего пользователя, либо 0, если пользователь неавторизирован), либо с помощью функции php **is_object** ().

Пример:

Вызвать некий метод класса.

```
if ( $AUTH_USER_ID) {
  $perm->метод();
}
```

или

```
".( is_object($perm) && $perm->метод() ? "А" : "Б" )."
```

Класс позволяет получить права не только текущего пользователя, но и любого другого пользователя, а так же группы.
Для этого в конструктор нужно передать номер пользователя или группы. Номер пользователя передается первым параметром, номер группы — вторым. Второй параметр необязателен.

Примеры:

Создать объект прав для пользователя с номером 5:
```
$perm_user_5 = new Permission( 5 );
```

Создать объект прав для группы с номером 2:

```
$perm_group = new Permission (0, 2);
```

Методы класса

Ниже приведён список методов, которые можно использовать.

```
bool isDirector()
```

Метод проверяет, есть ли у пользователя (группы) тип прав «Директор». Если пользователь — директор, то метод вернет true, иначе — false.

Пример использования:

Вывести в макете особое приветствие для пользователей-директоров.

```
".opt( is_object($perm) && $perm->isDirector(), "Вы являетесь
директором на этом сайте.")."
```

```
bool isSupervisor()
```

Метод проверяет, обладает ли пользователь правом «Супервизор» или выше. Если пользователь — директор или супервизор, то метод вернет true, иначе — false.

Пример использования:

Вывести в макете ссылку для входа в систему администрирования для супервизора и директора.

```
".opt( is_object($perm) && $perm->isSupervisor(), "<a
href='/netcat/admin/'>Войти в панель администратора</a>")."
```

```
bool isCatalogue($CatalogueID, $mask)
```

Метод проверяет, если у пользователя право, заданной маской $mask, на сайт с номером $CatalogueID

Параметры:

- **$CatalogueID** — номер сайта
- **$mask** — право, существование которого проверяется.

Метод возвращает true, если пользователь является директором, супервизором или редактором с требуемым правом.

$mask легче всего записывать через константы, приведённые выше.

Примеры использования:

Вывести сообщение в макете, если пользователь может добавлять и изменять свои объекты на сайте.

```
".opt( is_object($perm) && $perm->isCatalogue($catalogue,
MASK_ADD & MASK_EDIT), "Вы можете добавлять и изменять
сообщения")."
```

Данный текст пользователь увидит, если он:
- директор
- супервизор
- редактор всех сайтов с правами «добавление» и «изменение»
- редактор сайта с правами «добавление» и «изменение»

```
bool isSubdivision($SubdivisionID, $mask)
```

Метод проверяет, если у пользователя право, заданной маской $mask, на раздел с номером $SubdivisionID. Причем право пользователю может быть назначено на непосредственно не сам раздел, а на родительский раздел или на сайт, где находится раздел.
Параметры:
- **$SubdivisionID** — номер сайта
- **$mask** — право, существование которого проверяется.

Метод возвращает true, если пользователь является директором, супервизором или редактором с требуемым правом.
$mask можно записывать как и в случае с isCatalogue

Пример использования:

для модератора раздела при выводе объектов показать IP адрес пользователя, добавившего объект.
В системных настройках компонента вводим переменню $is_moderate, которая принимает значение true, если пользователь — модертор раздела и false — иначе.

```
$is_moderate = is_object($perm) && $perm->isSubdivision($sub,
MASK_MODERATE);
```

В объекте в списке выводим, если нужно, IP:

```
".opt( $is_moderate,  "IP: ".$f_IP)."
```

IP пользователь увидит, если он:

- директор
- супервизор
- редактор всех сайтов с правом «модерирование»
- редактор сайта, где находится этот раздел, с правом «модерирование»
- редактор этого раздела или раздела-родителя, с правом модерирование

```
bool isSubClass($SubClassID, $mask)
```

Метод проверяет, если у пользователя право, заданной маской $mask, на компонент в разделе с номером $SubClassID. Этот метод является очень важным в классе Permission, поскольку проверяет права непосредственно на компонент в разеделе.
Параметры:

- **$SubClassID** — номер сайта
- **$mask** — право, существование которого проверяется.

Метод возвращает true, если пользователь является директором, супервизором или редактором с требуемым правом.
$mask можно записывать, как и в случае с isCatalogue

Пример использования:

1. Узнать, является ли пользователь администратором компонента в разделе с номером 7:

```
( is_object($perm) && $perm->isSubClass(7, MASK_ADMIN) )
```

2. Вывести в префиксе компонента ссылку на добавление объекта, если пользователь действительно имеет право на добавление.

Здесь возможны три случая права доступа:

1. добавление разрешено всем
2. добавление разрешено зарегистрированным
3. добавление разрешено уполномоченным

Тип доступа содержится в элементе массива $cc_env по ключу Write_Access_ID
В первом случае добавление возможно всегда.
Во втором случае, только если существует переменная AUTH_USER_ID
В третьему случае, если существует объект $perm и isSubClass выдает true.
В префиксе можно написать:

```
".( ($cc_env['Write_Access_ID'] == 1 ||
    ($cc_env['Write_Access_ID'] == 2 && $AUTH_USER_ID) ||
```

```
    ( $cc_env['Write_Access_ID'] == 3 && is_object($perm) &&
$perm->isSubClass($cc, MASK_ADD | MASK_MODERATE) ) )? "<a
href='$addLink'>Добавить объект</a>" : "")."
```

nc_ImageTransform

Доступность: везде, при подключении соответствующего файла

Класс предназначен для работы с изображениями и содержит набор статических методов.

Подключение

По умолчанию этот класс не загружается. Чтобы его загрузить, нужно подключить файл *nc_imagetransform.class.php.*
Сделать это можно следующем способом:

```
require_once($INCLUDE_FOLDER."classes/nc_imagetransform.class.php"
);
```

$INCLUDE_FOLDER, если она не определена. перед подключением должна быть объявлена как глобальная.

Методы класса

string **imgResize($src_img**, **$dest_img**, **$width**, **$height**, **$mode**=0, **$format**='jpg', **$quality**=90, **$message_id** = 0, **$field** = 0**)**

Публичный статический метод служит для изменения размера изображения.
Параметры:

- **$src_img** — путь к исходному изображению;
- **$dest_img** — путь к создаваемому изображению;
- **$width** — ширина создаваемого изображения;
- **$height** — высота создаваемого изображения:
- **$mode** (необязательный) — режим уменьшения:
  - 0 — пропорционально уменьшает (по умолчанию);
  - 1 — вписывает в указанные размеры.
- **$format** (необязательный) — формат изображения: jpg, gif или png.
- **$quality** (необязательный) — качество сжатия изображения при формате «jpg». Может принимать значения от 0 до 100. По умолчанию — 90.

- **$message_id** (необязательный) — номер объекта, к которому относится файл.
- **$field** (необязательный) — номер поля или его имя, к которому относится файл.

Последние два параметра нужны при изменение файлов объектов или пользователей для обновления записей в таблицах.

Примеры использования:

Уменьшать аватар пользователя (поле **ForumAvatar**) до размеров 120 на 120.
В действиях после добавления/изменения пользователя нужно написать:

```
"; // разрываем html-код для вставки php-кода
// подключаем класс
require_once($INCLUDE_FOLDER."classes/nc_imagetransform.class.php"
);
// определяем текущий аватар
$photo_path    =    $DOCUMENT_ROOT.nc_file_path('User',    $message,
'ForumAvatar', "");
// при наличии аватара
if ( $photo_path) {
     // вызываем статический метод класса
     nc_ImageTransform::imgResize($photo_path,$photo_path,'120',
'120', 0, 'jpg', 90, $message, 'ForumAvatar');
}
echo "
```

Изменить размер изображения объекта. Имя поля — Photo:
В действиях после добавления нужно написать:

```
";
require_once($INCLUDE_FOLDER."classes/nc_imagetransform.class.php"
);

$photo_path   =   $DOCUMENT_ROOT.nc_file_path($classID,   $message,
'Photo', "");

if ( $photo_path) {
     nc_ImageTransform::imgResize($photo_path,$photo_path,'120',
'120', 0, 'jpg', 90, $message, 'Photo');
}
echo "
```

bool **createThumb($src_img**, **$dest_img**, **$width**, **$height**, **$mode**=0, **$format**='jpg', **$quality**=90, **$message_id** = 0, **$field** = 0**)**

Публичный статический метод для создания копии изображении с возможностью изменения ее размеров (создание превью).
Параметры:

- **$src_field_name** — имя поля с исходным изображением;
- $dest_field_name— имя поля приемника;
- **$width** — ширина создаваемого изображения;
- **$height** — высота создаваемого изображения:
- **$mode** (необязательный) — режим уменьшения:
  - 0 — пропорционально уменьшает (по умолчанию);
  - 1 — вписывает в указанные размеры.
- **$format** (необязательный) — формат изображения: jpg, gif или png.
- **$quality** (необязательный) — качество сжатия изображения при формате «jpg». Может принимать значения от 0 до 100. По умолчанию — 90.

Пример использования:

При добавлении объекта с изображением (например, фотогалерея) создать превью размером 50 на 50.
Имя поля с изображением — **Picture.**
Имя поля для превью — **Preview**.
В действие после добавления объекта нужно прописать:

```
";
   // если файл был закачан
  if ($_FILES['f_Picture'][size] != 0 ) {
    require_once($INCLUDE_FOLDER."classes/nc_imagetransform.class.
php");
    // создать превью
    nc_ImageTransform::createThumb('Picture','Preview',50,50);
  }
echo "
```