+7 (495) 223-46-50
+7 (812) 448-38-90
+7 (8636) 237-836
www.hostcms.ru
info@hostcms.ru
support@hostcms.ru

HostCMS — удобство управления сайтом в любой точке мира.

Система управления сайтом HostCMS v. 5

Руководство по разработке модулей

2011

Содержание

Средства разработки

Разработку модулей для HostCMS мы рекомендуем вести с использованием следующих программных продуктов:

- IDE (среда разработки программного обеспечения):
  - Eclipse PDT (PHP Development Tools Project), http://www.eclipse.org/pdt/
  - PHPEclipse, http://www.phpeclipse.com/
  - Notepad++, http://notepad-plus-plus.org/
- Браузер
  - Mozilla Firefox 3.x, http://www.mozilla-europe.org/ru/firefox/
    i. Средство отладки Firebug, http://getfirebug.com/

Кодировка файлов

Файлы модуля должны быть сохранены в кодировке UTF-8. Обратите внимание, использовать обычный Блокнот для этого нельзя, т.к. в начало будет добавляться BOM-метка (byte order mark, метка порядка байтов), что приведет к неработоспособности модуля. При работе с Notepad++ необходимо установить кодировку через меню Кодировки → Кодировать в UTF-8 без BOM, при работе с другими IDE установите кодировку UTF-8 в параметрах проекта.

Общая информация

Все модули системы располагаются в директории **/modules/**. Новый модуль располагается в поддиректории с именем нового модуля, например **entity**.

Внутри созданной директории размещается основной файл модуля с именем **[имя_модуля].php**, например **entity.php**.

Основной файл модуля

Основной файл модуля должен содержать некоторый набор обязательных директив:

```php
<?php
/**
 * Система управления сайтом HostCMS v. 5.xx
 * Copyright © 2005-2010 ООО "Хостмэйк" (Hostmake LLC), http://www.hostcms.ru
 *
 * Модуль entity.
 *
 * Файл: /modules/entity/entity.php
 *
 * @author Hostmake LLC
 * @version 5.x
 */

/* Путь к модулю */
$module_path_name = 'entity';

/* Имя модуля */
$module_name = 'Демонстрационный модуль';

// Указание соответствия имени класса и модуля
$GLOBALS['HOSTCMS_CLASS'][$module_path_name] = $module_path_name;
```

```
$kernel = & singleton('kernel');

/* Список файлов для загрузки */
$kernel->AddModuleFile($module_path_name, CMS_FOLDER .
"modules/{$module_path_name}/{$module_path_name}.class.php");

// Добавляем версию модуля
$kernel->add_modules_version($module_path_name, '5.x', '04.08.2010');

// Если раздел администрирования
if (defined('IS_ADMIN_PART'))
{
	// Подключаем языковой файл
	$kernel->LoadModulesLngFile($module_path_name, $module_name);

	// Помещаем модуль в меню
	$AdminMenu = new AdminMenu();
	$AdminMenu->AddAdminMenuItem(270, $module_name,
	"DoLoadAjax('/admin/{$module_path_name}/{$module_path_name}.php', '', 0, 'load_data', 0,
0, 0, 0);
	return false;", "/admin/{$module_path_name}/{$module_path_name}.php", $module_path_name,
3);
}?>
```

Особенности методов

AddModuleFile()

В метод AddModuleFile() необходимо передавать третий параметр, указывающий наименование класса, содержащегося в файле модуля. Этот параметр необходимо указывать при отличии наименование класса от наименования модуля, например:

```
$kernel->AddModuleFile($module_path_name, CMS_FOLDER .
"modules/{$module_path_name}/{$module_path_name}.class.php", 'mynewclass');
```

или

```
$kernel->AddModuleFile($module_path_name, CMS_FOLDER .
"modules/{$module_path_name}/mynewclass.class.php", 'mynewclass');
```

AddAdminMenuItem()

У метода AddAdminMenuItem() значение 270 указывает порядок сортировки пункта меню, значение 3 указывает блок правого меню, в котором располагается ссылка (счет ведется с 0). Более подробное описание методов можно найти в API HostCMS.

Динамическая загрузка модулей

При работы системы под управлением PHP-5 используется *__autoload()* для динамической загрузки модулей.

При использовании собственных модулей системе управления неизвестны соответствия классов и модулей, в связи с чем необходимо явно указать принадлежность классов модулям. Задание соответствия производится в суперглобальном массиве $GLOBALS['HOSTCMS_CLASS'] по следующей

схеме: ['class_name'] = 'module_name', при этом имя класса должно указываться в нижнем регистре, например для класса entity и модуля entity:

```
$GLOBALS['HOSTCMS_CLASS']['entity'] = 'entity';
```

Соответствия указываются в файле *main_classes.php*, который вызывается до загрузки ядра.

Класс модуля

Пример класса модуля, размещаемого в **/modules/[имя_модуля]/[имя_модуля].class.php**

```
<?php

class entity
{

	/**
	 * Массив групп сущностей
	 *
	 * @var array
	 * @access private
	 */
	var $entity_groups_array = array();

	/**
	 * Массив групп сущностей с иерархией
	 *
	 * @var array
	 * @access private
	 */
	var $masgroup = array();

	/**
	 * Вставка сущности
	 *
	 * @param $param array массив атрибутов
	 * - $param['entity_id'] int идентификатор сущности
	 * - $param['entity_group_id'] int идентификатор группы сущности
	 * - $param['entity_name'] string имя сущности
	 * - $param['entity_description'] string описание сущности
	 * - $param['entity_order'] int значение порядка сортировки
	 * - $param['users_id'] int идентификатор пользователя центра администрирования
	 * @return mixed возвращает идентификатор вставленной/обновлённой записи или false в
случае ошибки
	 */
	function InsertEntity($param)
	{
		// ...
	}

	/**
	 * Вставка группы сущностей
	 *
	 * @param $param array массив атрибутов
	 * - $param['entity_group_id'] int идентификатор группы сущностей
	 * - $param['entity_group_parent_id'] int идентификатор родительской группы
	 * - $param['entity_group_name'] string имя группы
	 * - $param['entity_group_description'] string описание группы
	 * - $param['entity_group_order'] int значение порядка сортировки
	 * - $param['users_id'] int идентификатор пользователя центра администрирования
```

```
         * @return mixed возвращает идентификатор вставленной/обновлённой записи или false в
случае ошибки
         */
        function InsertEntityGroup($param)
        {
                // ...
        }

        /**
         * Удаление сущности
         *
         * @param $entity_id int идентификатор сущности
         * @return boolean
         */
        function DeleteEntity($entity_id)
        {
                // ...
        }

        /**
         * Получение пути от текущей группы сущностей к корневой
         * @param int $entity_group_id Идентификатор группы сущностей
         * @param boolean $first_call флаг, указывающий на первый вызов функции
         * @return array массив данных
         */
        function GetEntityGroupPathArray($entity_group_id, $first_call = true)
        {
                // ...
        }

        /**
         * Удаление группы сущностей
         *
         * @param $entity_group_id int идентификатор группы сущностей
         * @return boolean
         */
        function DeleteEntityGroup($entity_group_id)
        {
                // ...
        }

        /**
         * Получаем список всех дочерних групп группы сущностей
         * @param int $entity_group_id идентификатор родительской группы
         * @return resource Результат запроса
         */
        function GetAllChildGroups($entity_group_id)
        {
                // ...
        }

        /**
         * Получение списка сущностей конкретной группы сущностей
         * @param int $entity_group_id
         * @return Resource результат выполнения запроса
         */
        function GetAllGroupEntities($entity_group_id)
        {
                // ...
        }

        /**
         * Получение информации о сущности
         *
         * @param int $entity_id идентификатор сущности
```

```
		 * @return mixed массив с данными о сущности или false, если сущность не найдена
		 */
		function GetEntity($entity_id)
		{
			// ...
		}

		/**
		 * Получение информации о группе сущностей
		 *
		 * @param int $entity_group_id идентификатор группы сущностей
		 * @return mixed array массив с данными о группе сущностей или false, если группа
сущностей не найдена
		 */
		function GetEntityGroup($entity_group_id)
		{
			// ...
		}

		/**
		 * Генерация дерева групп сущностей
		 *
		 * @param int $entity_group_id идентификатор группы с которой начинается генерация, если
0, то начинать с корневой
		 * @param boolean $first_call флаг, указывающий на первый вызов функции
		 * @param str $separator строка, используя которую функция отобразит вложенность одной
группы относительно другой
		 * @return array массив групп
		 */
		function GenGroupTree($entity_group_id, $first_call = true, $separator = '')
		{
			// ...
		}

		/**
		 * Получение списка всех групп сущностей
		 * @return Resource результат запроса к базе
		 */
		function GetAllEntityGroups()
		{
			// ...
		}

		/**
		 * Установка модуля
		 */
		function Install()
		{
			// ...
		}

		/**
		 * Удаление модуля
		 */
		function UnInstall()
		{
			// ...
		}
}
?>
```

Внимание! Создание класса модуля обязательно. Полный код класса можно посмотреть в файлах демонстрационного модуля.

В поддиректории **/modules/[имя_модуля]/config/** размещаются настройки модуля, доступные для редактирования пользователю системы. Настройки модуля могут содержать PHP-код и доступны при редактировании модуля из центра администрирования.

Внимание! Размещение просто текста в настройках модуля недопустимо, т.к. настройки модуля подключаются до генерации страницы и вывод текста может повредить работе центра администрирования и клиентского раздела.

Внимание! Созданный модуль необходимо добавить в список модулей через центр администрирования в разделе «Модули». Пока модуль не разработан, статус активности рекомендуется устанавливать в неактивное состояние. Если после включения модуля система перестала работать, посмотрите журнал событий системы управления в директории /logs/. Выключить модуль можно через базу данных в таблице *modules_table*.

Языковые файлы

В поддиректории **/modules/[имя_модуля]/lng/** размещаются языковые файлы. Файл размещается в директории с именем языка, например **/modules/[имя_модуля]/lng/ru/ru.php**

```php
<?php
/**
 * Система управления сайтом HostCMS v. 5.xx
 *
 * Copyright © 2005-2010 ООО "Хостмэйк" (Hostmake LLC), http://www.hostcms.ru
 *
 * Модуль: entity.
 *
 * Файл: /modules/entity/lng/ru/ru.php
 *
 * @author Hostmake LLC
 * @version 5.x
 */
$GLOBALS['MSG_entity'] = array();

$GLOBALS['MSG_entity']['notes'] = 'Заголовок блока';

$GLOBALS['MSG_entity']['entity_form_title'] = 'Управление сущностями';
$GLOBALS['MSG_entity']['entity_title'] = 'Управление группами сущностей';
$GLOBALS['MSG_entity']['entity_menu1'] = 'Меню для доступа к формам редактирования';
$GLOBALS['MSG_entity']['entity_path_message1'] = 'Управление группами сущностей';
$GLOBALS['MSG_entity']['entity_path_message2'] = 'Управление сущностями';
$GLOBALS['MSG_entity']['entity_group_add'] = 'Добавить';
$GLOBALS['MSG_entity']['entity_add'] = 'Добавить';
$GLOBALS['MSG_entity']['entity_group_del'] = 'Удалить группу';

$GLOBALS['MSG_entity']['entity_add'] = 'Добавить сущность';
$GLOBALS['MSG_entity']['entity_del'] = 'Удалить сущность';

// ...
// Полную версию файла можно посмотреть в демо-модуле на http://www.hostcms.ru/documentation/

?>
```

Язык по умолчанию для системы управления устанавливается в константе DEFAULT_LNG при установке, если константа не определена, используется значение *ru*.

Программирование раздела администрирования

Создается основной файл раздела администрирования модуля, имеющий путь */admin/[имя_модуля]/[имя_модуля].php*

```
<?php

// Подключаем основные классы.
require_once('../../main_classes.php');

$admin = new Admin();

// Проверка авторизации пользователя.
$admin->admin_session_valid('entity');

$kernel = & singleton('kernel');

// Инсталляция всех модулей ядра.
$kernel->LoadModules(!isset($_REQUEST['JsHttpRequest']));

// Инициализация CURRENT_SITE и констант.
$admin->admin_init();

if (!isset($_REQUEST['JsHttpRequest']))
{
	do_html_header_admin($GLOBALS['MSG_entity']['entity_form_title']);
}
$admin_forms = new admin_forms();

$entity = new entity();

// Обработчик кнопки "Применить"
if (isset($_REQUEST['entity_group_apply'])
|| isset($_REQUEST['entity_group_save']))
{
	ob_start();
	if (defined('READ_ONLY') && READ_ONLY)
	{
		$kernel->ReadMode();
	}
	else
	{
		// Добавляем информацию в БД.
		include(CMS_FOLDER . 'admin/entity/action/add_entity_group.php');

	}

	$message = ob_get_clean();
	$JsHttpRequest = new JsHttpRequest('UTF-8');

	// При добавлени (не сохранении) - отправляем команду на загрузку
	if (isset($_REQUEST['entity_group_apply']))
	{
		// Отображаем форму
		include(CMS_FOLDER . '/admin/entity/action/load_entity.php');
		$GLOBALS['_RESULT'] = $admin_forms->ShowForm(100000, 'load_data');

		// Добавляем определенное выше сообщение
		$GLOBALS['_RESULT']['error'] = $message;
	}
	else
	{
		// Отправляем только сообщение о произведенных действиях
```

```
                $GLOBALS['_RESULT'] = array(
                        'error' => $message,
                        'form_html' => '',
                        'title' => '');
        }
        echo $JsHttpRequest->LOADER;
        exit();

}

// Обработчик кнопок "Сохранить" и "Применить" формы "Редактирования/Добавления сущности"
if (isset($_REQUEST['entity_apply']) || isset($_REQUEST['entity_save']))
{
        ob_start();

        if (defined('READ_ONLY') && READ_ONLY)
        {
                $kernel->ReadMode();
        }
        else
        {
                // Добавляем информацию в БД.
                include(CMS_FOLDER . 'admin/entity/action/add_entity.php');
        }

        $message = ob_get_clean();
        $JsHttpRequest = new JsHttpRequest('UTF-8');

        // При добавлени (не сохранении) - отправляем команду на загрузку
        if (isset($_REQUEST['entity_apply']))
        {
                // Отображаем форму
                include(CMS_FOLDER . '/admin/entity/action/load_entity.php');
                $GLOBALS['_RESULT'] = $admin_forms->ShowForm(100000, 'load_data');

                // Добавляем определенное выше сообщение
                $GLOBALS['_RESULT']['error'] = $message;
        }
        else
        {
                // Отправляем только сообщение о произведенных действиях
                $GLOBALS['_RESULT'] = array(
                        'error' => $message,
                        'form_html' => '',
                        'title' => '');
        }
        echo $JsHttpRequest->LOADER;
        exit();
}

// Отображение форм
switch (to_int($_REQUEST['admin_forms_id']))
{
        case 100000:
        default:
                {
                        include(CMS_FOLDER . '/admin/entity/action/load_entity.php');
                        $admin_forms->ProcessAjax(100000);
                        break;
                }
        /*
        // Пример для других форм
        case 100001:
                {
                        include(CMS_FOLDER . '/admin/entity/action/load_myform2.php');
```

```
                        $admin_forms->ProcessAjax(to_int($_REQUEST['admin_forms_id']));
                        break;
                }
        case 100002:
                {
                        include(CMS_FOLDER.'/admin/entity/action/load_myform3.php');
                        $admin_forms->ProcessAjax(to_int($_REQUEST['admin_forms_id']));
                        break;
                }
        */
}

// Отображение "Подвала" сайта
if (!isset($_REQUEST['JsHttpRequest']))
{
        do_html_footer_admin();
}
?>
```

Создается директория */admin/[имя_модуля]/action/*, в которой будут размещаться файлы центра администрирования модуля.

Защита директории /admin/[имя_модуля]/action/

В директории */admin/[имя_модуля]/action/* необходимо создать файл «.htaccess» следующего содержания:

```
deny from all
```

Инициализация файлов модуля

Инициализация файлов модуля, включая языковые константы, для PHP-5 осуществляется при первом создании объекта класса, соответствующего модулю. Для этого в основной файл раздела администрирования перед обработкой заполненных форм помещается создание объекта класса, например:

```
$entity = new entity();
```

Формы центра администрирования

Формы центра администрирования и поля списочных форм добавляются через центр администрирования. Формы центра администрирования состоят из двух типов форм, описание которых дано в руководстве пользователя в разделе «*Интерфейс центра администрирования*».

Модуль может отображать несколько форм, при этом если идентификатор формы не был передан, используется значение по умолчанию (в данном случае ID такой формы = 100000). Идентификаторы форм можно получить из списка форм центра администрирования.

Выбор файлов для отображения форм размещается в основном файле раздела администрирования модуля, имеющий путь */admin/[имя_модуля]/[имя_модуля].php* (см. полный код файла выше).

```
// Отображение форм
switch (to_int($_REQUEST['admin_forms_id']))
{
```

```
        case 100000:
        default:
                {
                        include(CMS_FOLDER . '/admin/entity/action/load_entity.php');
                        $admin_forms->ProcessAjax(100000);
                        break;
                }
        // Пример для других форм
        case 100001:
                {
                        include(CMS_FOLDER . '/admin/entity/action/load_myform2.php');
                        $admin_forms->ProcessAjax(to_int($_REQUEST['admin_forms_id']));
                        break;
                }
        case 100002:
                {
                        include(CMS_FOLDER.'/admin/entity/action/load_myform3.php');
                        $admin_forms->ProcessAjax(to_int($_REQUEST['admin_forms_id']));
                        break;
                }
}
```

Обратите внимание, что для формы «по умолчанию» в метод *ProcessAjax* идентификатор формы передается в явном виде ввиду отсутствия в этом случае *$_REQUEST['admin_forms_id']*:

```
$admin_forms->ProcessAjax(100000);
```

Рассмотрим пример обработчика формы на примере файла *load_entity.php*:

```
<?php
/**
 * Ajax загрузчик для списка раздела сущностей и сущностей.
 * Модуль: /admin/entity/action/load_entity.php
 */
function LoadAjaxData(&$admin_forms)
{
        $on_page = $admin_forms->GetOnPageCount();

        $page = to_int($_REQUEST['limit']);

        $begin = ($page == 0) ? 0 : ($page - 1) * $on_page;

        $param = array();

        $param['limit']['all'] = array('begin' => $begin, 'count' => $on_page);

        $site_id = CURRENT_SITE;

        // получаем идентификатор родительской группы для реализации перехода по "хлебным крошкам"
        $entity_group_parent_id = to_int($_REQUEST['entity_group_parent_id']);

        // Строим сроку навигации
        $param['path_array'] = array();

        // Первый элемент массива позволяет вернуться в корневую группу.
        // Здесь передается entity_group_parent_id как дополнительный параметр, он равен "0".
        $param['path_array'][] = array('link' => $admin_forms->GetHtmlCallDoLoadAjax(
        $admin_forms->AAction, "&entity_group_parent_id=0", 100000, 'load_data'),
        'onclick' => $admin_forms->GetOnClickCallDoLoadAjax(
        $admin_forms->AAction, "&entity_group_parent_id=0", 100000, 'load_data'),
```

```
        'name' => $GLOBALS['MSG_entity']['entity_path_message1']);

        // получаем идентификатор экземпляра класса "entity"
        $entity = & singleton('entity');

        // строим хлебные крошки
        $path_array = $entity->GetEntityGroupPathArray($entity_group_parent_id, false);

        if ($path_array)
        {
                foreach ($path_array as $key => $value)
                {
                        // все последующие элементы массива "хлебных крошек" строятся автоматически
                        $param['path_array'][] = array (
                        'link' => $admin_forms->GetHtmlCallDoLoadAjax($admin_forms-
>AAction,"&entity_group_parent_id={$key}", 100000, 'load_data'),
                        'onclick' =>  $admin_forms->GetOnClickCallDoLoadAjax($admin_forms->AAction,
"&entity_group_parent_id={$key}", 100000, 'load_data'),
                        'name' => $value);
                }
        }

        // Запрос для получения количества элементов, запись вида "count(*) as count" -
ОБЯЗАТЕЛЬНА.
        // Запрос вернет количество элементов 0-го уровня (в данном случае групп)
        // ограничение по {ORDER_STRING} {LIMIT} для подсчета элементов не производится
        $count_entity_group = "SELECT count(*) as count
        FROM `entity_group_table` WHERE `site_id` = '{$site_id}'
        AND `entity_group_parent_id` = '{$entity_group_parent_id}'
        {FILTER_STRING}";

        // Запрос для получения непосредственно элементов (групп элементов)
        $param['data'][] = "SELECT entity_group_id as code, entity_group_name as name,
entity_group_order as order_field
        FROM `entity_group_table`
        WHERE `site_id` = '{$site_id}' AND `entity_group_parent_id` = '{$entity_group_parent_id}'
        {FILTER_STRING} {ORDER_STRING} {LIMIT}";

        // Запрос для получения количества элементов, запись вида "count(*) as count" -
ОБЯЗАТЕЛЬНА.
        // Запрос вернет количество элементов 1-го уровня (чаще всего элементов)
        $count_entity = "SELECT count(*) as count
        FROM `entity_table`
        WHERE `site_id` = '{$site_id}' AND `entity_group_id` = '{$entity_group_parent_id}'
        {FILTER_STRING}";

        // Запрос для получения непосредственно элементов 0-го уровня (чаще всего групп элементов)
        $param['data'][] = "SELECT entity_id as code, entity_name as name, entity_order as
order_field
        FROM `entity_table`
        WHERE `site_id` = '{$site_id}' AND `entity_group_id` = '{$entity_group_parent_id}'
        {FILTER_STRING} {ORDER_STRING} {LIMIT}";

        // Формируем верхнее меню.
        $k = 0;

        $menu = array();
        $menu[$k]['name'] = $GLOBALS['MSG_entity']['entity_path_message1'];
        $menu[$k]['link'] = '';
        $menu[$k]['icon'] = '';

        // Формируем субменю
        $sub_menu = array();
        $sub_menu[$p = 0]['name'] = $GLOBALS['MSG_entity']['entity_group_add'];
```

```
	$sub_menu[$p]['link'] = $admin_forms-
>GetHtmlCallTrigerSingleAction('entity_group_add_edit', 'check_0_0', 100000, 0, 0);
	$sub_menu[$p]['onclick'] = $admin_forms-
>GetOnClickCallTrigerSingleAction('entity_group_add_edit', 'check_0_0', 100000, 0, 0);
	$sub_menu[$p]['icon'] = '/admin/images/folder_add.gif';

	// Добавляем субменю элементу меню
	$menu[$k]['sub_items'] = $sub_menu;

	// Добавляем еще одно меню
	$menu[++$k]['name'] = $GLOBALS['MSG_entity']['entity_path_message2'];
	$menu[$k]['link'] = '';
	$menu[$k]['icon'] = '';

	$p = 0;

	// Добавляем новое субменю
	$sub_menu = array();

	$sub_menu[$p]['name'] = $GLOBALS['MSG_entity']['entity_add']= 'Добавить';
	$sub_menu[$p]['link'] = $admin_forms-
>GetHtmlCallTrigerSingleAction('entity_group_add_edit', 'check_1_0', 100000, 0, 0);
	$sub_menu[$p]['onclick'] = $admin_forms-
>GetOnClickCallTrigerSingleAction('entity_group_add_edit', 'check_1_0', 100000, 0, 0);
	$sub_menu[$p]['icon'] = '/admin/images/page_add.gif';

	// Добавляем подэлементы.
	$menu[$k]['sub_items'] = $sub_menu;

	$menu_array = array();
	$menu_array[] = $menu;

	// Для первого набора данных указываем общее количество
	$param['current_page'] = $page;
	$param['on_page'] = $on_page;
	$param['menus'] = $menu_array;

	// Заголовок формы
	$param['title'] = $GLOBALS['MSG_entity']['entity_title'];

	/*
		Переопределение свойств полей выводимой таблицы.
		$param['field_params'][X][Y][Z],
		    где X - целое число, означающее источник данных, Чаще всего 0 - это группы
элементов, 1 - непосредственно элементы;
			Y - ключевое поле;
			Z - специальный параметр, список всех параметров доступен в API.
	*/
	$param['field_params'][0]['img']['admin_forms_field_image'] = "=/admin/images/folder.gif";

	// Пример переопределения ссылки с названия для элементов
	// Для групп используется ссылка, для элементов ссылка не нужна
	$param['field_params'][1]['name']['admin_forms_field_type'] = 10;// вычисляемое поле
	$param['field_params'][1]['name']['callback_function'] = 'callback_function_name';

	// Функция обратного вызова для названия. Здесь пользователь может изменить формат
выводимых данных
	function callback_function_name($field_value, $value, $row)
	{
		echo $row['name'];
	}

	// Количество выводимых на страницу групп элементов
	$param['total_count'][] = $count_entity_group;
```

```
	// Количество выводимых на страницу элементов
	$param['total_count'][] = $count_entity;

	return $param;
}

// Необязательная функция для определения реального имени столбца по псевдониму,
// используется при нескольких источниках данных с разными именами полей
function ResolveFieldName(&$field_name, $dataset_id)
{
	switch ($dataset_id)
	{
		case 0: // список групп элементов
			{
				if ($field_name == 'code')
				{
					$field_name = 'entity_group_id';
				}
				elseif ($field_name == 'name')
				{
					$field_name = 'entity_group_name';
				}
				elseif ($field_name == 'order_field')
				{
					$field_name = 'entity_group_order';
				}
				break;
			}
		case 1: // список элементов
			{
				if ($field_name == 'code')
				{
					$field_name = 'entity_id';
				}
				elseif ($field_name == 'name')
				{
					$field_name = 'entity_name';
				}
				elseif ($field_name == 'order_field')
				{
					$field_name = 'entity_order';
				}
				break;
			}
	}
	return true;
}

// Обработчики действий

/**
 * Редактирование
 *
 * @param array $param
 */
function entity_group_add_edit(&$param)
{
	// ...
}

/**
 * Применить изменения
 *
 * @param array $param
 */
```

```
function apply_changes(&$param)
{
	// ...
}

/**
 * Удаление
 *
 * @param array $param
 */

function entity_delete(&$param)
{
	// ...
}
?>
```

Форма может иметь один или несколько источников данных. Например, вывод списка групп сущностей – это один источник данных, вывод сущностей– второй источник. Источники нумеруются с нуля.

Ограничение источников данных

Ограничения источников задаются тремя способами:

Способ № 1 — «Ограничение одного источника»:

```
	$param['limit'][0] = array('begin' => $begin, 'count' => $on_page);
```

Способ № 2 — «Ограничение двух источников с выводом на первой странице всех элементом первого и ограничением вывода второго»

```
	$param['limit'][0] = array();
	$param['limit'][1] = array('begin' => $begin, 'count' => $on_page);
```

Способ № 3 — «Ограничение двух источников с ограничением вывода первого и второго, при этом на страницу будут выводиться по *$on_page* элементов первого источника до момента, при котором к выводу из первого источника останется меньше, чем *$on_page* элементов, после чего выведутся недостающие до *$on_page* элементы второго источника и дальше будут выводиться элементы второго источника.»

```
	$param['limit']['all'] = array('begin' => $begin, 'count' => $on_page);
```

Количество элементов в источниках

Указывается SQL-запрос в переменной и передается следующим образом:

```
	// Для первого набора данных указываем общее количество
	$param['total_count'][] = $count;
	// Для второго набора данных указываем общее количество (если есть)
	$param['total_count'][] = $count2;
	// и т.д.
```

Если источник не является SQL-запросом, то в $count указывается целое значение.

ResolveFieldName() — определения реального имени столбца по псевдониму

Функция ResolveFieldName(&$field_name, $dataset_id), указываемая в обработчике загрузки формы, используется для определения реального имени столбца по псевдониму.

Наиболее часто функция используется для возможности осуществления фильтрации при выводе данных из 2-х и более источников, т.к. в таком случае имена результирующих столбцов в SQL-запросах указываются через псевдонимы.

```
// Необязательная функция для определения реального имени столбца по псевдониму,
// используется при нескольких источниках данных с разными именами полей
function ResolveFieldName(&$field_name, $dataset_id)
{
	switch ($dataset_id)
	{
		case 0: // список групп элементов
			{
				if ($field_name == 'code')
				{
					$field_name = 'entity_group_id';
				}
				elseif ($field_name == 'name')
				{
					$field_name = 'entity_group_name';
				}
				elseif ($field_name == 'order_field')
				{
					$field_name = 'entity_group_order';
				}
				break;
			}
		case 1: // список элементов
			{
				if ($field_name == 'code')
				{
					$field_name = 'entity_id';
				}
				elseif ($field_name == 'name')
				{
					$field_name = 'entity_name';
				}
				elseif ($field_name == 'order_field')
				{
					$field_name = 'entity_order';
				}
				break;
			}
	}
	return true;
}
```

Функции обратного вызова для полей формы

Пользовательские функции обратного вызова используются при необходимости разработки алгоритма особого вывода данных в ячейку форму. Сфера применения весьма обширна и позволяет создавать практически любое отображение ячейки.

Функция указывается и определяется внутри LoadAjaxData(), рассмотрим пример:

```
function LoadAjaxData(&$admin_forms)
{
	// ...

	// Заголовок формы
	$param['title'] = $GLOBALS['MSG_entity']['entity_title'];

	// Переопределяем отображение поля пользовательской функцией
	$param['field_params'][0]['file_image']['callback_function'] =
'callback_function_file_image';

	// Функция обратного вызова для отображениия изображений файлов и директорий
	function callback_function_file_image($field_value, $value, $row)
	{
		$kernel = & singleton('kernel');

		// Файл
		if ($row['fs_size'] != '<DIR>')
		{
			// Ассоциированные иконки
			$kernel = & singleton('kernel');
			$icon_array = $kernel->icon_array;

			// Определяем иконку файла
			$ext = strtolower($kernel->GetExtension($row['fs_name']));

			if (isset($icon_array[$ext]))
			{
				$icon_file = "/admin/images/icons/".to_str($icon_array[$ext]);
			}
			else
			{
				$icon_file = "/admin/images/icons/file.gif";
			}
		}
		else // Директория
		{
			$icon_file = "/admin/images/folder.gif";
		}
		?>
		<img src="<?php echo $icon_file;?>" />
		<?php
	}

	return $param;
}
```

В приведенном примере функция обратного вызова осуществляет вывод пиктограммы директории, если поле *fs_size* имеет значение '<DIR>', в противном случае определяется расширение файла по его имени из поля *$row['fs_name']* и поиск соответствующей пиктограммы.

Указание функции обратного вызова осуществляется через переопределение $param['field_params'] по следующей схеме:

```
$param['field_params'][{ID-источника}][{Имя_поля}]['callback_function'] = '{Имя_функции}';
```

Пример указания функции обратного вызова callback_function_file_image() для поля «**file_image**» первого источника:

```
$param['field_params'][0]['file_image']['callback_function'] = 'callback_function_file_image';
```

Функция обратного вызова обязательно принимает три параметра, при разработке рекомендуется распечатать все три массива для получения списка данных, которыми можно оперировать:

```
function callback_function_file_image($field_value, $value, $row)
{
	var_dump($field_value);
	var_dump($value);
	var_dump($row);
}
```

Действия форм центра администрирования

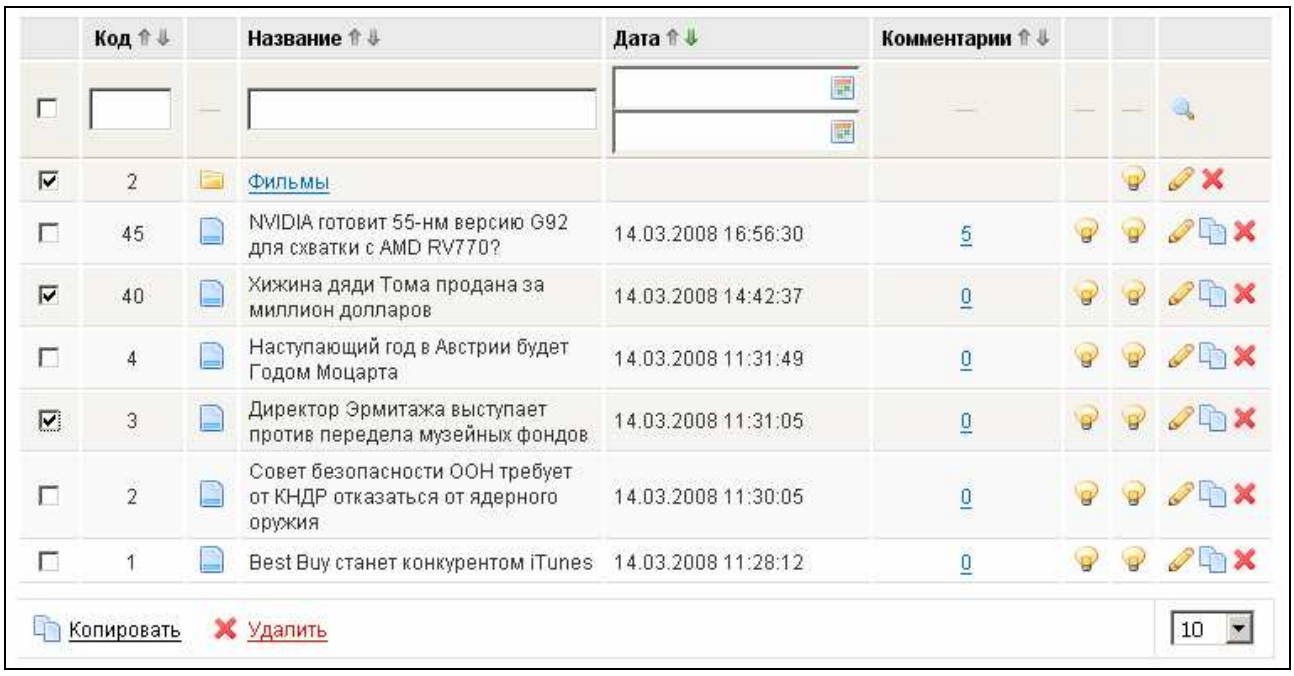
Список действий указывается для формы через центр администрирования, при этом для действия указывается имя функции-обработчика этого действия. Обработчики действий указываются в файле обработчика формы.

Пример указания обработчиков трех типовых действий – редактирование/создание, удаление и копирование.

```
<?php
/**
 * Ajax загрузчик для списка раздела сущностей и сущностей.
 * Модуль: /admin/entity/action/load_entity.php
 */
function LoadAjaxData(&$admin_forms)
{
	// ...

	// Функция обратного вызова для названия. Здесь пользователь может изменить формат
выводимых данных
	function callback_function_name($field_value, $value, $row)
	{
		// ...
	}

	// ...
}

// Необязательная функция для определения реального имени столбца по псевдониму,
// используется при нескольких источниках данных с разными именами полей
function ResolveFieldName(&$field_name, $dataset_id)
{
	// ...
}

// Обработчики действий

/**
 * Редактирование
 *
 * @param array $param
 */
function entity_group_add_edit(&$param)
{
	// ...
}
```

```
/**
 * Применить изменения
 *
 * @param array $param
 */
function apply_changes(&$param)
{
	// ...
}

/**
 * Удаление
 *
 * @param array $param
 */

function entity_delete(&$param)
{
	// ...
}
?>
```

Обработчик принимает массив идентификаторов выбранных строк для каждого источника. Рассмотрим пример групповой операции с элементами и группами информационных систем, в котором для первого источника (группы) будет передан идентификатор 2, для второго источника (элементы) будет передан идентификатор 40 и 3.



В случае, если обработчик действия должен вернуть данные для вывода, обработчик действия должен вернуть *true*, например, действие редактирования:

```
/**
 * Редактирование
 *
 * @param array $param
 */
function entity_group_add_edit(&$param)
{
```

```
        ob_start();

        // ...

        $param['result'] = ob_get_clean();

        return true;
}
```

Если действие выводит только сообщение о своем результате, возвращать ничего не нужно:

```
/**
 * Применить изменения
 *
 * @param array $param
 */
function apply_changes(&$param)
{
        ob_start();

        // ...

        $error = ob_get_clean();
        $param['error'] = $error;
}


 * Удаление
 *
 * @param array $param
 */

function entity_delete(&$param)
{
        ob_start();

        // ...

        $error = ob_get_clean();
        $param['error'] = $error;
}
```

Обработчик редактирования

Обратите внимание, $param['data'][номер_источника] содержит массив выбранных в клиентском разделе элементов (при единичной или групповой операции). Массив $param['data'][0] в данном случае не имеет ничего общего с аналогичным массивом при задании SQL-запросов для источников данных.

```
/**
 * Редактирование
 *
 * @param array $param
 */
function entity_group_add_edit(&$param)
{
        ob_start();

        // Определяем экземпляры необходимых объектов
```

```
	$user_access = & singleton('user_access');

	$entity = & singleton('entity');

	if (isset($param['data']))
	{
		if (isset($param['data'][0]))
		{
			foreach ($param['data'][0] as $entity_group_id => $value)
			{
				// Определяем режим доступа пользователя и принадлежит ли объект
пользователю
				$row = $entity->GetEntityGroup($entity_group_id);

				if($row)
				{
					$user_id = $row['users_id'];
				}
				else
				{
					$user_id = 0;
				}

				if($user_access->IssetUserAccessForObject($user_id))
				{

	include(CMS_FOLDER.'admin/entity/action/form_entity_group_add_edit.php');

					break;
				}
				else
				{
					$user_access->MessageOnlyOwn();
					$param['error'] = ob_get_clean();
					return ;
				}
			}

			// Устанавливаем <title> для страницы
			$param['title'] = ($entity_group_id == 0)
			? $GLOBALS['MSG_entity']['form_entity_group_add_title']
			: $GLOBALS['MSG_entity']['form_entity_group_edit_title'];
		}
		if (isset($param['data'][1]))
		{
			foreach ($param['data'][1] as $entity_id => $value)
			{
				// Определяем режим доступа пользователя и принадлежит ли объект
пользователю
				$row = $entity->GetEntity($entity_id);

				if($row)
				{
					$user_id = $row['users_id'];
				}
				else
				{
					$user_id = 0;
				}

				if($user_access->IssetUserAccessForObject($user_id))
				{

	include(CMS_FOLDER.'admin/entity/action/form_entity_add_edit.php');
					break;
```

```
                        }
                        else
                        {
                                $user_access->MessageOnlyOwn();
                                $param['error'] = ob_get_clean();
                                return ;
                        }

                        include(CMS_FOLDER .
'admin/entity/action/form_entity_add_edit.php');
                        break;
                }

                $param['title'] = ($entity_id == 0)
                ? $GLOBALS['MSG_entity']['form_entity_add_title']
                : $GLOBALS['MSG_entity']['form_entity_edit_title'];
        }
    }

    $param['result'] = ob_get_clean();

    return true;
}
```

Результат работы обработчика (форма редактирования) передается в $param['result'],

$param может принимать значения:

1. $param['result'] — информация для размещения в основном рабочем поле;
2. $param['error'] — информация для размещения в поле сообщений (размещается над основным полем);
3. $param['title'] — заголовок страницы (значение тэга <title>).

Обратите внимание на break в foreach(), в случае редактирования он необходим, т.к. редактирование не является групповым действием. Количество проверок на наличие $param['data'][{Номер-источника}] равно количеству источников для формы. Если источник один, функция будет иметь вид:

```
/**
 * Редактирование
 *
 * @param array $param
 */
function entity_group_add_edit(&$param)
{
    ob_start();

    // Определяем экземпляры необходимых объектов
    $user_access = & singleton('user_access');

    $entity = & singleton('entity');

    if (isset($param['data']))
    {
        if (isset($param['data'][0]))
        {
            foreach ($param['data'][0] as $entity_group_id => $value)
            {
                // Определяем режим доступа пользователя и принадлежит ли объект
пользователю
                $row = $entity->GetEntityGroup($entity_group_id);
```

```
                        if($row)
                        {
                                $user_id = $row['users_id'];
                        }
                        else
                        {
                                $user_id = 0;
                        }

                        if($user_access->IssetUserAccessForObject($user_id))
                        {

        include(CMS_FOLDER.'admin/entity/action/form_entity_group_add_edit.php');

                                break;
                        }
                        else
                        {
                                $user_access->MessageOnlyOwn();
                                $param['error'] = ob_get_clean();
                                return ;
                        }
                }

                // Устанавливаем <title> для страницы
                $param['title'] = ($entity_group_id == 0)
                ? $GLOBALS['MSG_entity']['form_entity_group_add_title']
                : $GLOBALS['MSG_entity']['form_entity_group_edit_title'];
        }
    }

    $param['result'] = ob_get_clean();

    return true;
}
```

Форма редактирования

Пример формы редактирования в файле action/form_entity_add_edit.php:

```
<?php
/*
 Форма редактирования/добавления сущности
 Модуль: /admin/entity/action/form_edit_add_entity.php
 Здесь определяется как будет выглядеть форма
 */

$form_params = array();

if(!isset($entity_id))
{
    $entity_id = 0;
}

$form_params['title'] = ($entity_id == 0)
? $GLOBALS['MSG_entity']['form_entity_add_title']
: $GLOBALS['MSG_entity']['form_entity_edit_title'];

// Метод отправки данных
$form_params['form_attribs']['method'] = 'post';
```

```
// Адрес всех обработчиков модуля – главный файл формы
$form_params['form_attribs']['action'] = '/admin/entity/entity.php';

//Имя формы
$form_params['form_attribs']['name'] = 'EditAddEntity';

$admin_forms_fields = new admin_forms_fields();
// Массив для хранения строки навигации
$form_params['path_array'] = array();

$entity = & singleton('entity');

// Указывается строка навигации, аналогично обработчику формы на примере файла load_myform1.php
// В данном случае при построении строки навигации вместо $admin_forms используется
$admin_forms_fields
$entity_group_parent_id = to_int($_REQUEST['entity_group_parent_id']);

// формируем хлебные крошки
$path_array = $entity->GetEntityGroupPathArray($entity_group_parent_id, false);

$form_params['path_array'][] = array(
	'link' => $admin_forms_fields->GetHtmlCallDoLoadAjax(
		$admin_forms_fields->AAction,"&entity_group_parent_id=0",
		100000, 'load_data'),
	'onclick' => $admin_forms_fields->GetOnClickCallDoLoadAjax(
		$admin_forms_fields->AAction,
		"&entity_group_parent_id=0", 100000, 'load_data'),
	'name' => $GLOBALS['MSG_entity']['entity_path_message1']);

if ($path_array)
{
	foreach ($path_array as $key => $value)
	{
		$form_params['path_array'][] = array (
			'link' => $admin_forms_fields->GetHtmlCallDoLoadAjax(
				$admin_forms_fields->AAction,
				"&entity_group_parent_id={$key}",
				100000,
				'load_data'),
			'onclick' =>  $admin_forms_fields->GetOnClickCallDoLoadAjax(
				$admin_forms_fields->AAction,
				"&entity_group_parent_id={$key}",
				100000,
				'load_data'),
			'name' => $value);
	}
}

$admin_forms_fields->admin_forms_fields($form_params);

if ($entity_id)
{
	$row = $entity->GetEntity($entity_id);

	$entity_id = to_int($row['entity_id']);

	$site_id = $row['site_id'];
}
else
{
	$site_id = CURRENT_SITE;
}

// Первая закладка
```

```
$tab_id = $admin_forms_fields-
>AddTab($GLOBALS['MSG_entity']['form_entity_group_add_edit_name_tab1']);

// Количество элементов на страницу.
$param = array();
$param['tab_id'] = $tab_id;
$param['name'] = 'entity_name';
$param['caption'] = $GLOBALS['MSG_entity']['entity_name'];
$param['type'] = 0; // Поле ввода.

if ($entity_id)
{
	$param['value'] = to_str($row['entity_name']);
}

$param['attributes']['class'] = 'large';

$param['format']['minlen']['value'] = 1;
$param['format']['maxlen']['value'] = 255;

$admin_forms_fields->AddField($param);

// Разделитель.
$admin_forms_fields->AddSeparator($tab_id);

// формируем поле выбора родительской группы
$parent_array_groups = array();

$parent_array_groups[0] = ' ... ';

$temp_array = $entity->GenGroupTree(0, "     ");

// заполнение массива родителских групп
$temp_array_count = count($temp_array);

for ($i = 0; $i < $temp_array_count; $i++)
{
	if(isset($temp_array[$i]['entity_group_id']))
	{
		// текущая группа не должна попасть в список, она не может быть родительской самой
себе
		if (isset($entity_id)
		&& ($entity_id == to_int($temp_array[$i]['entity_group_id'])))
		{
			continue;
		}
		$parent_array_groups[$temp_array[$i]['entity_group_id']] =
$temp_array[$i]['separator'].$temp_array[$i]['entity_group_name'];
	}
}

$param = array();
$param['tab_id'] = $tab_id;
$param['name'] = 'entity_group_id';
$param['caption'] = $GLOBALS['MSG_entity']['entity_parent_group'];
$param['type'] = 2; // Выпадающий список
$param['items'] = $parent_array_groups;
$param['value'] = $entity_group_parent_id;

$admin_forms_fields->AddField($param);

// Разделитель.
$admin_forms_fields->AddSeparator($tab_id);

$param = array();
```

```
$param['tab_id'] = $tab_id;
$param['name'] = 'entity_order';
$param['caption'] = $GLOBALS['MSG_entity']['entity_order'];
$param['type'] = 0; // Поле ввода.

if ($entity_id)
{
	$param['value'] = to_str($row['entity_order']);
}

$admin_forms_fields->AddField($param);

// Разделитель.
$admin_forms_fields->AddSeparator($tab_id);

if ($entity_id)
{
	// Добавляем скрытое поле с идентификатором сущности, т.к. идет редактирование.
	$param = array();
	$param['tab_id'] = $tab_id;
	$param['name'] = 'entity_id';
	$param['type'] = 4; // Скрытое поле.
	$param['value'] = to_int($row['entity_id']);
	$admin_forms_fields->AddField($param);
}

$tab_id = $admin_forms_fields-
>AddTab($GLOBALS['MSG_entity']['form_entity_group_add_edit_name_tab2']);

$param = array();
$param['tab_id'] = $tab_id;
$param['name'] = 'entity_description';
$param['caption'] = $GLOBALS['MSG_entity']['entity_description'];
$param['type'] = 5; // Поле ввода.

if ($entity_id)
{
	$param['value'] = to_str($row['entity_description']);
}

$admin_forms_fields->AddField($param);

// Кнопка "Сохранить".
$param = array();
$param['name'] = 'entity_save';
$param['caption'] = $GLOBALS['MSG_entity']['entity_group_save'];
$param['attributes']['onclick'] = "return SendForm('{$admin_forms_fields->GetWindowId()}',
'{$admin_forms_fields->AAction}',
'{$admin_forms_fields->AAdditionalParams}', this, ".to_int($_REQUEST['admin_forms_id']).",
'load_data', " . to_int($admin_forms_fields->form_params['current_page']) . ", " .
to_int($admin_forms_fields->form_params['on_page']) . " )";
$param['attributes']['type'] = 'button';
$admin_forms_fields->AddButton($param);

// Кнопка "Применить".
$param = array();
$param['name'] = 'entity_apply';
$param['caption'] = $GLOBALS['MSG_entity']['entity_group_apply'];
$param['image'] = '/admin/images/add.gif';
$param['attributes']['onclick'] = "return SendForm('{$admin_forms_fields->GetWindowId()}',
'{$admin_forms_fields->AAction}',
'{$admin_forms_fields->AAdditionalParams}', this, ".to_int($_REQUEST['admin_forms_id']).",
'load_data', " . to_int($admin_forms_fields->form_params['current_page']) . ", " .
to_int($admin_forms_fields->form_params['on_page']) . " )";
$param['attributes']['type'] = 'submit';
```

```
$admin_forms_fields->AddButton($param);

// Отображение формы.
$admin_forms_fields->ShowForm();
?>
```

Подробное описание метода AddField($param) дано в API.

Обработка заполненной формы осуществляется в основном файле.

Обратите внимание, при добавления списков с множественным выбором вместо *$param['name'] = 'parent_field_id';* необходимо использовать *$param['name'] = 'parent_field_id**[]**';*

Внимание! При редактировании сущности необходимо обязательно добавлять скрытое поле, содержащее идентификатор редактируемой записи.

Вызов действия добавления из меню

Добавление сущности является частным случаем редактирования, при котором идентификатор сущности равен 0.

Вышеприведенный пример меню основной формы содержит вызов обработчика редактирования для добавления новой сущности:

```
       $sub_menu[$p]['name'] = $GLOBALS['MSG_entity']['entity_add']= 'Добавить';
       $sub_menu[$p]['link'] = $admin_forms-
>GetHtmlCallTrigerSingleAction('entity_group_add_edit', 'check_1_0', 100000, 0, 0);
       $sub_menu[$p]['onclick'] = $admin_forms-
>GetOnClickCallTrigerSingleAction('entity_group_add_edit', 'check_1_0', 100000, 0, 0);
       $sub_menu[$p]['icon'] = '/admin/images/page_add.gif';
```

Методы GetHtmlCallTrigerSingleAction() и GetOnClickCallTrigerSingleAction() имеют одинаковые атрибуты и служат для создания ссылок для вызова действий. В примере вызывается действие **«entity_group_add_edit»** формы **«100000»** поля с ID=0 второго источника — **«check_1_0»**. Если требуется вызвать добавление для первого источника, вместо **«check_1_0»** указывается **«check_0_0»**.

Обратите внимание, имя действия «entity_group_add_edit» является общим для первого (группы) и второго (элементы) источника.

Обработчик удаления

При удалении нет необходимости выводить информацию в основное рабочее поле, соответственно значение этого поля не передается и остается без изменения.

Удаление является критичным процессом и в режиме «Только чтение» не должно быть доступно.

```
/**
 * Удаление
 *
 * @param array $param
 */

function entity_delete(&$param)
{
```

```
	ob_start();

	// Определяем экземпляры необходимых объектов
	$user_access = & singleton('user_access');

	$entity = & singleton('entity');

	if (defined('READ_ONLY') && READ_ONLY)
	{
		$kernel = & singleton('kernel');
		$kernel->ReadMode();
	}
	else
	{
		$count = 0;

		if (isset($param['data'][0]))
		{
			foreach ($param['data'][0] as $key => $value)
			{
				// Определяем режим доступа пользователя и принадлежит ли объект
пользователю
				$row = $entity->GetEntityGroup($key);

				if($row)
				{
					$user_id = $row['users_id'];
				}
				else
				{
					$user_id = 0;
				}

				if($user_access->IssetUserAccessForObject($user_id))
				{
					if(!$entity->DeleteEntityGroup($key))
					{
						$flag_success = false;
					}
					else
					{
						$count++;

						$flag_success = true;
					}
				}
				else
				{
					$user_access->MessageOnlyOwn();
					$param['error'] = ob_get_clean();
					return ;
				}
			}

			if ($flag_success)
			{
				if ($count == 1)
				{

	show_message($GLOBALS['MSG_entity']['group_entity_delete_success']);
				}
				else
				{

	show_message($GLOBALS['MSG_entity']['groups_entity_delete_success']);
```

```
                                }
                        }
                }
                if (isset($param['data'][1]))
                {
                        foreach ($param['data'][1] as $key => $value)
                        {
                                // Определяем режим доступа пользователя и принадлежит ли объект
пользователю
                                $row = $entity->GetEntity($key);

                                if($row)
                                {
                                        $user_id = $row['users_id'];
                                }
                                else
                                {
                                        $user_id = 0;
                                }

                                if($user_access->IssetUserAccessForObject($user_id))
                                {
                                        if(!$entity->DeleteEntity($key))
                                        {
                                                $flag_success = false;
                                        }
                                        else
                                        {
                                                $count++;

                                                $flag_success = true;
                                        }
                                }
                                else
                                {
                                        $user_access->MessageOnlyOwn();
                                        $param['error'] = ob_get_clean();
                                        return ;
                                }
                        }

                        if ($flag_success)
                        {
                                if ($count == 1)
                                {

        show_message($GLOBALS['MSG_entity']['entity_delete_success']);
                                }
                                else
                                {

        show_message($GLOBALS['MSG_entity']['entities_delete_success']);
                                }
                        }
                }
        }

        $error = ob_get_clean();
        $param['error'] = $error;
}
```

Обработчик копирования и др.

Обработчик копирования и другие обработчики по своей структуре схожи с обработчиком удаления или обработчиком редактирования. Если не осуществляется вывод в рабочую область, то за основу рекомендуется брать пример обработчика удаления, в противном случае рекомендуется использовать пример обработчика редактирования.

Обработка заполненных форм

Обработка заполненных форм осуществляется в основном файле модуля центра администрирования:

```
// Обработчик кнопок "Сохранить" и "Применить" формы "Редактирования/Добавления сущности"
if (isset($_REQUEST['entity_apply']) || isset($_REQUEST['entity_save']))
{
	ob_start();

	if (defined('READ_ONLY') && READ_ONLY)
	{
		$kernel->ReadMode();
	}
	else
	{
		// Добавляем информацию в БД.
		include(CMS_FOLDER . 'admin/entity/action/add_entity.php');
	}

	$message = ob_get_clean();
	$JsHttpRequest = new JsHttpRequest('UTF-8');

	// При добавлени (не сохранении) - отправляем команду на загрузку
	if (isset($_REQUEST['entity_apply']))
	{
		// Отображаем форму
		include(CMS_FOLDER . '/admin/entity/action/load_entity.php');
		$GLOBALS['_RESULT'] = $admin_forms->ShowForm(100000, 'load_data');

		// Добавляем определенное выше сообщение
		$GLOBALS['_RESULT']['error'] = $message;
	}
	else
	{
		// Отправляем только сообщение о произведенных действиях
		$GLOBALS['_RESULT'] = array(
			'error' => $message,
			'form_html' => '',
			'title' => '');
	}
	echo $JsHttpRequest->LOADER;
	exit();
}
```

Внесение данных в базу данных

Внесение данных в базу данных выносится в отдельный файл, в данном случае это файл *action/add_entity.php*:

```
<?php
/*
 Обработчик добавления сущности.
```

```
 Модуль: /admin/entity/action/add_entity.php
 */

// Получаем переданные поля
$site_id = CURRENT_SITE;

// выбираем пользователя из сессии
if (isset($_SESSION["current_users_id"]))
{
	$users_id = to_int($_SESSION["current_users_id"]);
}
else
{
	$users_id = 0;
}

if(isset($_REQUEST['entity_id']))
{
	$entity_id = to_int($_REQUEST['entity_id']);
}
else
{
	$entity_id = 0;
}

// Если добавление прошло успешно, обязательно добавим на форму скрытое поле с ID добавленной
записи
// Не забываем указать имя формы, в данном случае это 'EditAddEntity'
if ($new_entity_id =
	$entity->InsertEntity(array(
	'entity_id' => $entity_id,
	'entity_group_id' => to_int($_REQUEST['entity_group_id']),
	'entity_name' => to_str($_REQUEST['entity_name']),
	'entity_order' => to_int($_REQUEST['entity_order']),
	'site_id' => $site_id,
	'entity_description' => to_str($_REQUEST['entity_description']))))
{

	$admin_forms = new admin_forms();
	$window_id = $admin_forms->GetWindowId();
	?><script type="text/javascript">
	$.appendInput('<?php echo $window_id?>', 'EditAddEntity', 'entity_id', '<?php echo
$new_entity_id?>');
	</script><?php

	// Редактирование сущности
	if ($entity_id)
	{
		show_message($GLOBALS['MSG_entity']['entity_edit_success']);
	}
	else // Добавление сущности
	{
		show_message($GLOBALS['MSG_entity']['entity_add_success']);
	}
}
else // Ошибка добавления/редактирования
{
	if ($entity_id)
	{
		show_error_message($GLOBALS['MSG_entity']['entity_edit_error']);
	}
	else
	{
		show_error_message($GLOBALS['MSG_entity']['entity_add_error']);
	}
```

```
}
?>
```

Добавление внешнего JavaScript файла для подключения в центре администрирования

В настройках модуля добавьте внешние JavaScript файлы, необходимые для его функционирования:

```
$kernel = & singleton('kernel');
$kernel->AddJs('/admin/js/my.js');
```

Установка модулей

Кроме самих файлов, модуль может содержать SQL-запросы и другие инструкции, которые необходимо выполнять при установке или удалении модуля из системы.

Для этой цели необходимо использовать методы Install() и UnInstall() класса, являющегося основным в модуле. Имена методов Install() и UnInstall() являются регистрозависимыми.

При добавлении модуля в систему проверяется наличие метода Install() у основного класса нового модуля и, если он доступен, указанный метод вызывается. При удалении модуля система аналогично поступает с методом UnInstall(). Обратите внимание, при отключенном модуле вызов Install() и UnInstall() не производится.

Вывод блока на главной странице центра администрирования

Для вывода блока на главной странице центра администрирования используется метод *AdminMainPage()* основного класса модуля. Если метод не объявлен, то блок для соответствующего модуля не выводится.

** Функционал доступен с версии 5.8.4.*

```
/**
 * Функция обратного вызова для отображения блока
 * на основной странице центра администрирования.
 *
 */
function AdminMainPage()
{
	// Получаем информацию о пользователе
	$user_access = & singleton('user_access');
	$user_row = $user_access->GetUser($_SESSION['current_users_id']);

	$access_row = GetUserAccess('entity', $user_row['users_type_id'], CURRENT_SITE);

	// Доступ к модулю разрешен или пользователь является суперюзером
	if (to_int($access_row['users_access_value']) == 1
	|| to_int($user_row['users_superuser']))
	{
		?>
		<td valign="top" class="index_table_td">
		<div class="main_div"><span class="div_title"><?php echo
$GLOBALS['MSG_entity']['notes']?></span>
			<div class="div_content">
			Содержимое блока
			</div>
		</div>
		</td>
		<?php

		return true;
	}

	return false;
}
```

Поисковая индексация модуля

Для индексации данных пользовательского модуля используется метод SearchIndexing*()* основного класса модуля. Если метод не объявлен, то поисковая индексация данных этого модуля не производится.

** Функционал доступен с версии 5.8.6.*

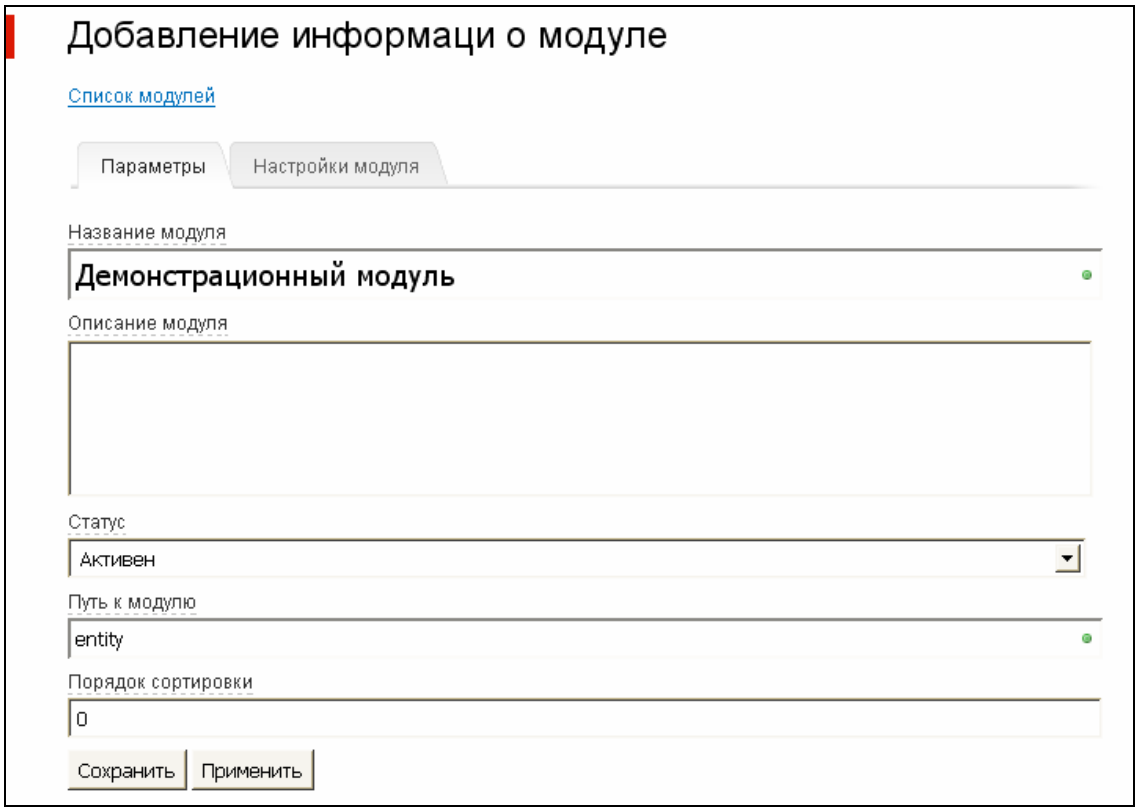
```
	/**
	 * Функция обратного вызова для поисковой индексации данных модуля
	 *
	 * @param $limit текущая позиция
	 * @param $on_step шаг
	 * @return array
	 */
	function SearchIndexing($limit, $on_step)
	{
		$result = $this->Indexation ($limit, $on_step);
		return $result;
	}
```

Метод индексации должен возвращать данные, подготовленные в соответствии с форматом входных данных метода Search:Insert_search_word().

Демонстрационный модуль

Загрузить демонстрационный модуль можно с сайта: http://www.hostcms.ru/documentation/

Загруженные файлы модуля разместите в системе управления, после чего добавьте модуль в список модулей. Все нужные таблицы и формы будут созданы автоматически.



Безопасность при разработке модулей

В HostCMS существует ряд методов и функций, предназначенных для приведения входных данных к типу, который ожидает программист.

Не забывайте проверять наличие элемента суперглобального массива перед его применением., например, если ожидается $_GET['my_id'], а пользователь этот параметр не передаст, то обращение к элементу создаст ошибку уровня Notice. Чтобы этого избежать проверяйте наличие элемента массива, например:

```
<?php
if (isset($_GET['my_id']))
{
echo to_int($_GET['my_id']);
}
?>
```

Функции приведения типа

Все внешние параметры должны приводиться к тому типу данных, который вы ожидаете. Под внешними параметрами подразумеваются данные из внешних источников, например: $_GET, $_POST, $_REQUEST, $_COOKIE, $_SERVER.

Используются следующие функции:

- to_str() — функция приведения аргумента к строковому типу;
- to_int() — функция приведения аргумента к целочисленному типу;
- to_float() — функция приведения аргумента к вещественному типу;
- to_bool() — функция приведения аргумента к логическому типу;
- to_array() — функция приведения аргумента к массиву.

Функции приведения типа имеют несколько особенностей:

1. Если переменная не существует, то ошибка «Notice: Undefined variable» не произойдет, при этом будет возвращено значение указанно типа, например, применение to_str() к несуществующей переменной вернет пустую строку, to_int() вернет ноль и т.д.
2. Если переменная не существовала, то она будет неявно объявлена. Эту особенность необходимо учитывать при использовании isset() над переменной, которая ранее был обработана указанными функциями.

Обратите внимание, указанные функции приведения типа можно использовать только с переменным, например:

```
$a = to_str($b);
$c = to_int($_REQUEST['myvalue']);
```

Неправильное использование:

```
$a = to_array(function($b)); // Функция применяется к результату другой функции
$c = to_float(MY_CONSTANT); // Функция применяется к константе
```

Работа с XML

Функция str_for_xml() преобразует строку к виду, пригодному для публикации в XML. Все внешние данные, которые добавляются в генерируемый XML должны быть обработаны этой функцией.

Работа с базой данных

Все внешние данные, помещаемые в запрос, должны быть обработаны функцией quote_smart(), при этом значения данных при размещении в запрос должны быть обрамлены апострофами.

Проверка прав доступа к модулю

В основном файле модуля в центре администрирования необходимо автоматически проверять права доступа пользователя к модулю. Проверка осуществляется методом *$admin->admin_session_valid()* с передачей ему имени модуля, например для модуля *entity*:

```
<?php

// Подключаем основные классы.
require_once('../../main_classes.php');

$admin = new Admin();

// Проверка авторизации пользователя.
$admin->admin_session_valid('entity');

$kernel = & singleton('kernel');

// Инсталляция всех модулей ядра.
$kernel->LoadModules(!isset($_REQUEST['JsHttpRequest']));

// Инициализация CURRENT_SITE и констант.
$admin->admin_init();

if (!isset($_REQUEST['JsHttpRequest']))
{
	do_html_header_admin($GLOBALS['MSG_entity']['entity_form_title']);
}
$admin_forms = new admin_forms();

$entity= new entity();
```

Константа IS_ADMIN_PART

Константа *IS_ADMIN_PART* существует и имеет значение *true* при работе в центре администрирования системы.

Если необходимо подключать языковые файлы только в центре администрирования, можно использовать следующую конструкцию:

```
<?php
/**
 * Система управления сайтом HostCMS v. 5.xx
 * Copyright © 2005-2010 ООО "Хостмэйк" (Hostmake LLC), http://www.hostcms.ru
 *
 * Модуль entity.
```

```
 *
 * Файл: /modules/entity/entity.php
 *
 * @author Hostmake LLC
 * @version 5.x
 */

// ...

// Если раздел администрирования
if (defined('IS_ADMIN_PART'))
{
	// Подключаем языковой файл
	$kernel->LoadModulesLngFile($module_path_name, $module_name);

	// Помещаем модуль в меню
	$AdminMenu = new AdminMenu();
	$AdminMenu->AddAdminMenuItem(270, $module_name,
	"DoLoadAjax('/admin/{$module_path_name}/{$module_path_name}.php', '', 0, 'load_data', 0,
0, 0, 0);
	return false;", "/admin/{$module_path_name}/{$module_path_name}.php", $module_path_name,
3);
}
?>
```

Особенности разработки при переходе на UTF-8

С версии 5.9.14 система управления HostCMS перешла на представление модулей в кодировке UTF-8, в связи с чем произошли следующие изменения:

1. Все файлы должны быть в кодировке UTF-8 без BOM.
2. AJAX-ответ всегда дается в UTF-8, для этого строки:

```
$JsHttpRequest = new JsHttpRequest(SITE_CODING);
```

заменяем на:

```
$JsHttpRequest = new JsHttpRequest('UTF-8');
```

3. Генерируемый XML необходимо всегда создавать в UTF-8.

Изменения в структуре модулей

Версия 5.9.23

1. Изменен принцип создания скрытого поля с идентификатором создаваемого объекта.

Было:

```
	<script>
	var ElementInput = document.createElement("input");
	ElementInput.setAttribute("type", "hidden");
	ElementInput.setAttribute("name", "entity_id");
	ElementInput.setAttribute("value", "<?php echo $new_entity_id?>");
	document.getElementById('EditAddEntity').appendChild(ElementInput);
	</script>
```

Стало:

```
	$admin_forms = new admin_forms();
	$window_id = $admin_forms->GetWindowId();
	?><script type="text/javascript">
	$.appendInput('<?php echo $window_id?>', 'EditAddEntity', 'entity_id', '<?php echo
$new_entity_id?>');
	</script><?php
```

2. В обработчиках кнопок форм JavaScript функция doSendForm() заменена на функцию SendForm()

Было:

```
$param['attributes']['onclick'] = "return doSendForm('{$admin_forms_fields->AAction}',
'{$admin_forms_fields->AAdditionalParams}', this, ".to_int($_REQUEST['admin_forms_id']).",
'load_data', " . to_int($admin_forms_fields->form_params['current_page']) . ", " .
to_int($admin_forms_fields->form_params['on_page']) . " )";
```

Стало:

```
$param['attributes']['onclick'] = "return SendForm('{$admin_forms_fields->GetWindowId()}',
'{$admin_forms_fields->AAction}',
'{$admin_forms_fields->AAdditionalParams}', this, ".to_int($_REQUEST['admin_forms_id']).",
'load_data', " . to_int($admin_forms_fields->form_params['current_page']) . ", " .
to_int($admin_forms_fields->form_params['on_page']) . " )";
```

Обратите внимание на добавление нового аргумента функции. Использование doSendForm() в версии 5.9.23 все еще возможно, однако мы настоятельно рекомендуем использовать новую функцию.