+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. 6

Руководство по разработке модулей

2013

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**
  - Firefox 22.0+, http://www.mozilla-europe.org/ru/firefox/
    i. Средство отладки Firebug, http://getfirebug.com/
  - IE 9+
  - Opera 12+, http://ru.opera.com/download/
  - Google Chrome 20+, https://www.google.com/intl/ru/chrome/browser/
  - Safari 5+, http://support.apple.com/kb/DL1531

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 IDE установите кодировку UTF-8 в параметрах проекта.

Демонстрационный модуль

В настоящем руководстве рассматривается создание модуля, содержащего некие структурированные системы (`entity`), каждая система содержит группы (`enity_group`) с неограниченным уровнем вложенности и элементы (`entity_item`). Все модули системы располагаются в директории `/modules/`, файлы центра администрирования в `/admin/`.

Структура файлов модуля

```
/admin/
/admin/entity/
/admin/entity/item/index.php
/admin/entity/index.php
/modules/
/modules/entity/
/modules/entity/config/config.php
/modules/entity/controller/
/modules/entity/controller/edit.php
/modules/entity/controller/show.php
/modules/entity/group/
/modules/entity/group/i18n/
/modules/entity/group/i18n/model.php
/modules/entity/i18n/
/modules/entity/item/controller/edit.php
/modules/entity/item/i18n/
/modules/entity/item/model.php
/modules/entity/model.php
/modules/entity/module.php
```

```
Центр администрирования
Центр администрирования модуля entity
Обработка показа entity_item и entity_group
Обработка показа entity
Модули
Модуль entity
Конфигурационный файл модуля
Контроллеры entity
Контроллер редактирования entity
Контроллер показа в клиентском разделе
Файлы, связанные с entity_group
Интернационализация entity_group
Модель entity_group
Интернационализация entity
Контроллер редактирования group и item
Интернационализация entity_item
Модель entity_item
Модель entity
Основной файл модуля
```

Общая информация

Основной файл модуля

Основной файл с данными о модуле `/modules/entity/module.php` должен содержать класс с набором обязательных директив. Класс наследуется от `Core_Module` и имеет имя, зависимое от названия модуля, например, `Entity_Module`.

```php
<?php

defined('HOSTCMS') || exit('HostCMS: access denied.');

/**
 * Entity.
 *
 * @package HostCMS 6\Entity
 * @version 6.x
 * @author Hostmake LLC
 * @copyright © 2005-2013 ООО "Хостмэйк" (Hostmake LLC), http://www.hostcms.ru
 */
class Entity_Module extends Core_Module
{
	/**
	 * Module version
	 * @var string
	 */
	public $version = '6.1';

	/**
	 * Module date
	 * @var date
	 */
	public $date = '2013-06-18';

	/**
	 * Constructor.
	 */
	public function __construct()
	{
		parent::__construct();

		$this->menu = array(
			array(
				'sorting' => 260,
				'block' => 1,
				'name' => Core::_('Entity.model_name'),
				'href' => "/admin/entity/index.php",
				'onclick' => "$.adminLoad({path: '/admin/entity/index.php'}); return
false"
			)
		);
	}

	/**
	 * Install module.
	 */
	public function install()
	{

	}
}
```

Класс модуля содержит публичные свойства `version` и `date` с данными о версии модуля и дате. Информация о пункте меню в центре администрирования содержится в публичном свойстве `menu`, представляющем собой массив пунктов меню.

Каждый элемент меню содержит:

- порядок сортировки пункта в блоке меню;
- номер блока в меню (для шаблона HostCMS 5), в котором располагается пункт меню (счет ведется с 0);
- название модуля в меню;
- ссылка на открытие модуля в новом окне;
- обработчик onclick.

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 `/hostcmsfiles/logs/`. Выключить модуль можно через базу данных в таблице `modules`.

Модель Entity

Модель `Entity_Model` является представлением таблицы `entities`. Пример модели, размещаемой в `/modules/[имя_модуля]/model.php`

```
<?php

defined('HOSTCMS') || exit('HostCMS: access denied.');

/**
 * Entity.
 *
 * @package HostCMS 6\Entity
 * @version 6.x
 * @author Hostmake LLC
 * @copyright © 2005-2013 ООО "Хостмэйк" (Hostmake LLC), http://www.hostcms.ru
 */
class Entity_Model extends Core_Entity
{
	/**
	 * Backend property
	 * @var mixed
	 */
	public $img = 1;

	/**
	 * One-to-many or many-to-many relations
	 * @var array
	 */
	protected $_hasMany = array(
		'entity_item' => array(),
		'entity_group' => array(),
	);

	/**
	 * Belongs to relations
	 * @var array
	 */
	protected $_belongsTo = array(
		'site' => array(),
		'structure' => array(),
```

```
	);

	/**
	 * Constructor.
	 */
	public function __construct($id = NULL)
	{
		parent::__construct($id);

		if (is_null($id))
		{
			$oUserCurrent = Core_Entity::factory('User', 0)->getCurrent();
			$this->_preloadValues['site_id'] = defined('CURRENT_SITE') ? CURRENT_SITE :
0;
		}
	}
}
```

Языковые файлы

Модель может иметь языковые файлы, размещаются они в `i18n` в директории модели. Файл имеет имя языка, например `/modules/[имя_модуля]/i18n/ru.php`

```
<?php

/**
 * Entity.
 *
 * @package HostCMS 6\Entity
 * @version 6.x
 * @author Hostmake LLC
 * @copyright © 2005-2013 ООО "Хостмэйк" (Hostmake LLC), http://www.hostcms.ru
 */
return array(
	'model_name' => 'Демонстрационный модуль',
	'menu' => 'Управление сущностями',
	'add' => 'Добавить',
	'edit_success' => "Сущность успешно изменена!",
	'edit_error' => "Ошибка добавления/редактирования сущности!",
	'edit_title' => 'Редактирование сущности',
	'add_title' => 'Добавление сущности',
	'id' => 'Идентификатор',
	'name' => 'Имя сущности',
	'site_id' => 'Идентификатор сайта',
	'site_name' => 'Сайт',
	'structure_name' => 'Узел структуры',
	'markDeleted_success' => 'Сущность успешно удалена!',
	'markDeleted_error' => 'Ошибка! Сущность не удалена!',
	'delete_success' => 'Элемент удален!',
	'undelete_success' => 'Элемент восстановлен!',
);
```

Программирование раздела администрирования

Основной файл раздела центра администрирования размещается в `/admin/[имя_модуля]/index.php`

```
<?php

/**
 * Entity.
 *
 * @package HostCMS
 * @version 6.x
 * @author Hostmake LLC
 * @copyright © 2005-2013 ООО "Хостмэйк" (Hostmake LLC), http://www.hostcms.ru
 */
require_once('../../bootstrap.php');
```

Проверка на авторизацию и права доступа к модулю entity.

```
Core_Auth::authorization('entity');
```

Получение объекта формы с кодом 100000.

```
// Код формы
$iAdmin_Form_Id = 100000;
$sAdminFormAction = '/admin/entity/index.php';

$oAdmin_Form = Core_Entity::factory('Admin_Form', $iAdmin_Form_Id);
```

Контроллер формы принимает в аргумент конструктора объект формы, устанавливаются путь к обработчику формы, заголовок и заголовок страницы.

```
// Контроллер формы
$oAdmin_Form_Controller = Admin_Form_Controller::create($oAdmin_Form);
$oAdmin_Form_Controller
	->setUp()
	->path($sAdminFormAction)
	->title(Core::_('Entity.model_name'))
	->pageTitle(Core::_('Entity.model_name'));
```

Меню формы центра администрирования создается с использованием фабрики `Admin_Form_Entity::factory('Menus')`, которой добавляются пункты меню с использованием `Admin_Form_Entity::factory('Menu')`.

```
// Меню формы
$oAdmin_Form_Entity_Menus = Admin_Form_Entity::factory('Menus');

// Элементы меню
$oAdmin_Form_Entity_Menus->add(
	Admin_Form_Entity::factory('Menu')
		->name(Core::_('Entity.menu'))
		->add(
			Core::factory('Admin_Form_Entity_Menu')
				->name(Core::_('Entity.add'))
```

```
                        ->img('/admin/images/page_add.gif')
                        ->href(
                                $oAdmin_Form_Controller-
>getAdminActionLoadHref($oAdmin_Form_Controller->getPath(), 'edit', NULL, 0, 0)
                        )
                        ->onclick(
                                $oAdmin_Form_Controller-
>getAdminActionLoadAjax($oAdmin_Form_Controller->getPath(), 'edit', NULL, 0, 0)
                        )
        )
);
```

Все меню, строка навигации и т.п. добавляется контроллеру формы с использованием метода `addEntity()`.

```
// Добавляем все меню контроллеру
$oAdmin_Form_Controller->addEntity($oAdmin_Form_Entity_Menus);
```

Строка навигации создается с использованием фабрики `Admin_Form_Entity::factory('Breadcrumbs')`, которой добавляются пункты меню с использованием `Admin_Form_Entity::factory('Breadcrumb')`.

```
// Элементы строки навигации
$oAdmin_Form_Entity_Breadcrumbs = Admin_Form_Entity::factory('Breadcrumbs');

// Элементы строки навигации
$oAdmin_Form_Entity_Breadcrumbs->add(
        Admin_Form_Entity::factory('Breadcrumb')
                ->name(Core::_('Entity.model_name'))
                ->href(
                        $oAdmin_Form_Controller->getAdminLoadHref($oAdmin_Form_Controller-
>getPath(), NULL, NULL, '')
                )
                ->onclick(
                        $oAdmin_Form_Controller->getAdminLoadAjax($oAdmin_Form_Controller-
>getPath(), NULL, NULL, '')
        )
);
```

В центре администрирования над объектами формы действия могут применяться двумя способами:

1. Метод модели объекта, например, для действия `changeActive` в модели объекта создается метод

```
public function changeActive()
        {
                $this->active = 1 - $this->active;
                $this->save();

                return $this;
        }
```

2. Внешний контроллер, например, добавление обработки события `edit`

```
// Действие редактирования
$oAdmin_Form_Action = Core_Entity::factory('Admin_Form', $iAdmin_Form_Id)
        ->Admin_Form_Actions
```

```
        ->getByName('edit');

if ($oAdmin_Form_Action && $oAdmin_Form_Controller->getAction() == 'edit')
{
        $oEntity_Controller_Edit = new Entity_Controller_Edit(
                $oAdmin_Form_Action
        );

        $oEntity_Controller_Edit
                ->addEntity($oAdmin_Form_Entity_Breadcrumbs);

        // Добавляем типовой контроллер редактирования контроллеру формы
        $oAdmin_Form_Controller->addAction($oEntity_Controller_Edit);
}
```

Каждая форма может иметь 1 или более источник данных, например список элементов, список групп и т.п.

```
// Источник данных 0
$oAdmin_Form_Dataset = new Admin_Form_Dataset_Entity(
        Core_Entity::factory('Entity')
);

// Ограничение источника 1 по родительской группе
$oAdmin_Form_Dataset->addCondition(
        array('where' =>
                array('site_id', '=', CURRENT_SITE)
        )
);

// Добавляем источник данных контроллеру формы
$oAdmin_Form_Controller->addDataset(
        $oAdmin_Form_Dataset
);
```

Обработка действий и показ формы.

```
// Показ формы
$oAdmin_Form_Controller->execute();
```

Формы центра администрирования

Формы центра администрирования и поля списочных форм добавляются через центр администрирования. Форма может иметь один или несколько источников данных. Например, вывод списка групп сущностей — нулевой источник данных, вывод сущностей — первый источник.

Ограничение источников данных

Ограничения источников задаются с помощью метода `addCondition()`, добавляемого к источнику:

```
// Ограничение источника 1 по родительской группе
$oAdmin_Form_Dataset->addCondition(
        array('where' =>
                array('site_id', '=', CURRENT_SITE)
        )
```

```
);
```

Метод `addCondition()` можно применять к источнику несколько раз, поддерживается method chaining.

Функции обратного вызова для полей формы

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

Функция указывается и определяется в соответствующих моделях, например:

```
	public function name()
	{
		$object = $this->shortcut_id
			? $this->Informationsystem_Item
			: $this;

		$oCore_Html_Entity_Div = Core::factory('Core_Html_Entity_Div')->value(
			htmlspecialchars($object->name)
		);

		$bRightTime = ($this->start_datetime == '0000-00-00 00:00:00' || time() >
Core_Date::sql2timestamp($this->start_datetime))
			&& ($this->end_datetime == '0000-00-00 00:00:00' || time() <
Core_Date::sql2timestamp($this->end_datetime));

		!$bRightTime && $oCore_Html_Entity_Div->class('wrongTime');

		// Зачеркнут в зависимости от своего статуса
		if (!$this->active)
		{
			$oCore_Html_Entity_Div->class('inactive');
		}
		elseif ($bRightTime)
		{
			$oCurrentAlias = $object->Informationsystem->Site->getCurrentAlias();

			if ($oCurrentAlias)
			{
				$href = 'http://' . $oCurrentAlias->name
					. $object->Informationsystem->Structure->getPath()
					. $object->getPath();

				$oCore_Html_Entity_Div
				->add(
					Core::factory('Core_Html_Entity_A')
						->href($href)
						->target('_blank')
						->add(
							Core::factory('Core_Html_Entity_Img')
							->src('/admin/images/new_window.gif')
							->class('img_line')
						)
				);
			}
		}
		elseif (!$bRightTime)
		{
			$oCore_Html_Entity_Div
				->add(
```

```
                                        Core::factory('Core_Html_Entity_Img')
                                                ->src('/admin/images/mesures.gif')
                                                ->class('img_line')
                                );
                }

                $oCore_Html_Entity_Div->execute();
        }
```

Действия форм центра администрирования

Список действий указывается для формы через центр администрирования, при этом для действия указывается имя функции-обработчика этого действия. Добавление обработчиков действий производится в файле `/admin/[имя_модуля]/index.php`

Пример указания обработчиков типовых действий `редактирование/создание` и `применение`.

```
// Действие редактирования
$oAdmin_Form_Action = Core_Entity::factory('Admin_Form', $iAdmin_Form_Id)
        ->Admin_Form_Actions
        ->getByName('edit');

if ($oAdmin_Form_Action && $oAdmin_Form_Controller->getAction() == 'edit')
{
        $oEntity_Controller_Edit = new Entity_Controller_Edit(
                $oAdmin_Form_Action
        );

        $oEntity_Controller_Edit
                ->addEntity($oAdmin_Form_Entity_Breadcrumbs);

        // Добавляем типовой контроллер редактирования контроллеру формы
        $oAdmin_Form_Controller->addAction($oEntity_Controller_Edit);
}

// Действие "Применить"
$oAdminFormActionApply = Core_Entity::factory('Admin_Form', $iAdmin_Form_Id)
        ->Admin_Form_Actions
        ->getByName('apply');

if ($oAdminFormActionApply && $oAdmin_Form_Controller->getAction() == 'apply')
{
        $oControllerApply = new Admin_Form_Action_Controller_Type_Apply
        (
                $oAdminFormActionApply
        );

        // Добавляем типовой контроллер редактирования контроллеру формы
        $oAdmin_Form_Controller->addAction($oControllerApply);
}
```

Типовой контроллер редактирования

Пример контроллера редактирования групп и элементов, размещенный в файле`/modules/[имя модуля]/item/controller/edit.php`

```
<?php
```

```
defined('HOSTCMS') || exit('HostCMS: access denied.');

/**
 * Entity item and group controller edit.
 *
 * @package HostCMS 6\Entity
 * @version 6.x
 * @author Hostmake LLC
 * @copyright © 2005-2013 ООО "Хостмэйк" (Hostmake LLC), http://www.hostcms.ru
 */
class Entity_Item_Controller_Edit extends Admin_Form_Action_Controller_Type_Edit
{
	/**
	 * Set object
	 * @param $object object
	 * @return self
	 */
	public function setObject($object)
	{
		$modelName = $object->getModelName();

		$entity_id = intval(Core_Array::getGet('entity_id', 0));
		$entity_group_id = Core_Array::getGet('entity_group_id', 0);

		switch($modelName)
		{
			case 'entity_item':

				if (is_null($object->id))
				{
					$object->entity_group_id = $entity_group_id;
					$object->entity_id = $entity_id;
				}

				parent::setObject($object);

				$oMainTab = $this->getTab('main');
				$oAdditionalTab = $this->getTab('additional');

				$oAdditionalTab->delete($this->getField('entity_group_id'));

				$oAdmin_Form_Entity_Select = new Admin_Form_Entity_Select();
				$oAdmin_Form_Entity_Select
					->name('entity_group_id')
					->options(
						array(' … ') + $this->fillEntityGroup($object->entity_id, 0)
					)
					->caption(Core::_('Entity_Item.entity_group_id'))
					->value($this->_object->entity_group_id);

				$oMainTab->addAfter($oAdmin_Form_Entity_Select, $this->getField('name'));

				$this->title(
					$this->_object->id
						? Core::_('Entity.edit_title')
						: Core::_('Entity.add_title')
					);
			break;
			case 'entity_group':
			default:

				if (is_null($object->id))
				{
```

```
                                $object->parent_id = $entity_group_id;
                                $object->entity_id = $entity_id;
                        }

                        parent::setObject($object);

                        $oMainTab = $this->getTab('main');
                        $oAdditionalTab = $this->getTab('additional');

                        $oAdditionalTab->delete($this->getField('parent_id'));

                        $oAdmin_Form_Entity_Select = new Admin_Form_Entity_Select();
                        $oAdmin_Form_Entity_Select
                                ->name('parent_id')
                                ->options(
                                        array(' … ') + $this->fillEntityGroup($object-
>entity_id, 0, array($this->_object->id))
                                )
                                ->caption(Core::_('Entity_Group.parent_id'))
                                ->value($this->_object->parent_id);

                        $oMainTab->addAfter($oAdmin_Form_Entity_Select, $this-
>getField('name'));

                        $this->title(
                                $this->_object->id
                                        ? Core::_('Entity_Group.edit_title')
                                        : Core::_('Entity_Group.add_title')
                        );

                break;
        }

        return $this;
    }

    /**
     * Entity groups tree
     * @var array
     */
    protected $_aGroupTree = array();

    /**
     * Build visual representation of group tree
     * @param int $iEntityId site ID
     * @param int $iEntityGroupParentId parent ID
     * @param int $aExclude exclude group ID
     * @param int $iLevel current nesting level
     * @return array
     */
    public function fillEntityGroup($iEntityId, $iEntityGroupParentId = 0, $aExclude =
array(), $iLevel = 0)
    {
        $iEntityId = intval($iEntityId);
        $iEntityGroupParentId = intval($iEntityGroupParentId);
        $iLevel = intval($iLevel);

        if ($iLevel == 0)
        {
            $aTmp = Core_QueryBuilder::select('id', 'parent_id', 'name')
                ->from('entity_groups')
                ->where('entity_id', '=', $iEntityId)
                ->where('deleted', '=', 0)
                ->orderBy('sorting')
                ->orderBy('name')
```

```
                        ->execute()->asAssoc()->result();

                foreach ($aTmp as $aGroup)
                {
                        $this->_aGroupTree[$aGroup['parent_id']][] = $aGroup;
                }
        }

        $aReturn = array();

        if (isset($this->_aGroupTree[$iEntityGroupParentId]))
        {
                $countExclude = count($aExclude);
                foreach ($this->_aGroupTree[$iEntityGroupParentId] as $childrenGroup)
                {
                        if ($countExclude == 0 || !in_array($childrenGroup['id'],
$aExclude))
                        {
                                $aReturn[$childrenGroup['id']] = str_repeat(' ', $iLevel) .
$childrenGroup['name'];
                                $aReturn += $this->fillEntityGroup($iEntityId,
$childrenGroup['id'], $aExclude, $iLevel + 1);
                        }
                }
        }

        $iLevel == 0 && $this->_aGroupTree = array();

        return $aReturn;
    }
}
```

При использовании нескольких источников данных разделение представления осуществляется в методе setObject() конструкцией

```
$modelName = $object->getModelName();

switch($modelName)
{
    case 'entity_item':

    // ...

    break;
    case 'entity_group':
    default:

    // ...

    break;
}
```

Обработка заполненной формы

Обработка заполненной формы контроллером редактирования, унаследованным от `Admin_Form_Action_Controller_Edit`, производится унаследованным методом `_applyObjectProperty()` и отдельного объявления не требует.

В случае необходимости реализации собственной логики обработки реализуете метод `_applyObjectProperty()` в своем контроллере.

```
	/**
	 * Processing of the form. Apply object fields.
	 */
	protected function _applyObjectProperty()
	{
		parent::_applyObjectProperty();

		if(
			// Поле файла существует
			!is_null($aFileData = Core_Array::getFiles('source', NULL))
			// и передан файл
			&& intval($aFileData['size']) > 0)
		{
			if (Core_File::isValidExtension($aFileData['name'], array('jpg', 'gif',
'png', 'swf')))
			{
				$this->_object->saveFile($aFileData['tmp_name'],
$aFileData['name']);
			}
			else
			{
				$this->addMessage(
					Core_Message::get(
						Core::_('Core.extension_does_not_allow',
Core_File::getExtension($aFileData['name'])),
						'error'
					)
				);
			}
		}
	}
```

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() основного класса модуля.

```
/**
 * Install module.
 */
public function install()
{
	$query = "Текст запроса";

	// Выполняем запрос
	Sql_Controller::instance()->execute($query);
}
```

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.

Поисковая индексация модуля

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

```
	/**
	 * Функция обратного вызова для поисковой индексации
	 *
	 * @param $offset
	 * @param $limit
	 * @return array
	 */
	public function indexing($offset, $limit)
	{
		/**
		 * $_SESSION['search_block'] - номер блока индексации
		 */
		if (!isset($_SESSION['search_block']))
		{
			$_SESSION['search_block'] = 0;
		}

		if (!isset($_SESSION['last_limit']))
		{
			$_SESSION['last_limit'] = 0;
		}

		$limit_orig = $limit;

		$result = array();

		switch ($_SESSION['search_block'])
		{
			case 0:
				$aTmpResult = $this->indexingEntityGroups($offset, $limit);

				$_SESSION['last_limit'] = count($aTmpResult);

				$result = array_merge($result, $aTmpResult);
				$count = count($result);

				if ($count < $limit_orig)
				{
					$_SESSION['search_block']++;
					$limit = $limit_orig - $count;
					$offset = 0;
				}
				else
				{
					return $result;
				}

			case 1:
				$aTmpResult = $this->indexingEntity($offset, $limit);

				$_SESSION['last_limit'] = count($aTmpResult);

				$result = array_merge($result, $aTmpResult);
				$count = count($result);

				// Закончена индексация
```

```
                        if ($count < $limit_orig)
                        {
                                $_SESSION['search_block']++;
                                $limit = $limit_orig - $count;
                                $offset = 0;
                        }
                        else
                        {
                                return $result;
                        }
                }

                $_SESSION['search_block'] = 0;

                return $result;
        }

        /**
         * Индексация групп сущностей
         *
         * @param int $offset
         * @param int $limit
         * @return array
         */
        public function indexingEntityGroups($offset, $limit)
        {
                $offset = intval($offset);
                $limit = intval($limit);

                $oEntityGroups = Core_Entity::factory('Entity_Group');
                $oEntityGroups
                        ->queryBuilder()
                        ->where('entity_groups.deleted', '=', 0)
                        ->limit($offset, $limit);

                $aEntityGroups = $oEntityGroups->findAll();

                $result = array();
                foreach($aEntityGroups as $oEntityGroup)
                {
                        $result[] = $oEntityGroup->indexing();
                }

                return $result;
        }

        /**
         * Индексация сущностей
         *
         * @param int $offset
         * @param int $limit
         * @return array
         */
        public function indexingEntity($offset, $limit)
        {
                $offset = intval($offset);
                $limit = intval($limit);

                $oEntities = Core_Entity::factory('Entity');

                $oEntities
                        ->queryBuilder()
                        ->where('entities.deleted', '=', 0)
                        ->limit($offset, $limit);
```

```
		$aEntities = $oEntities->findAll();

		$result = array();
		foreach($aEntities as $oEntity)
		{
			$result[] = $oEntity->indexing();
		}

		return $result;
	}
```

Функция обратного вызова для поиска

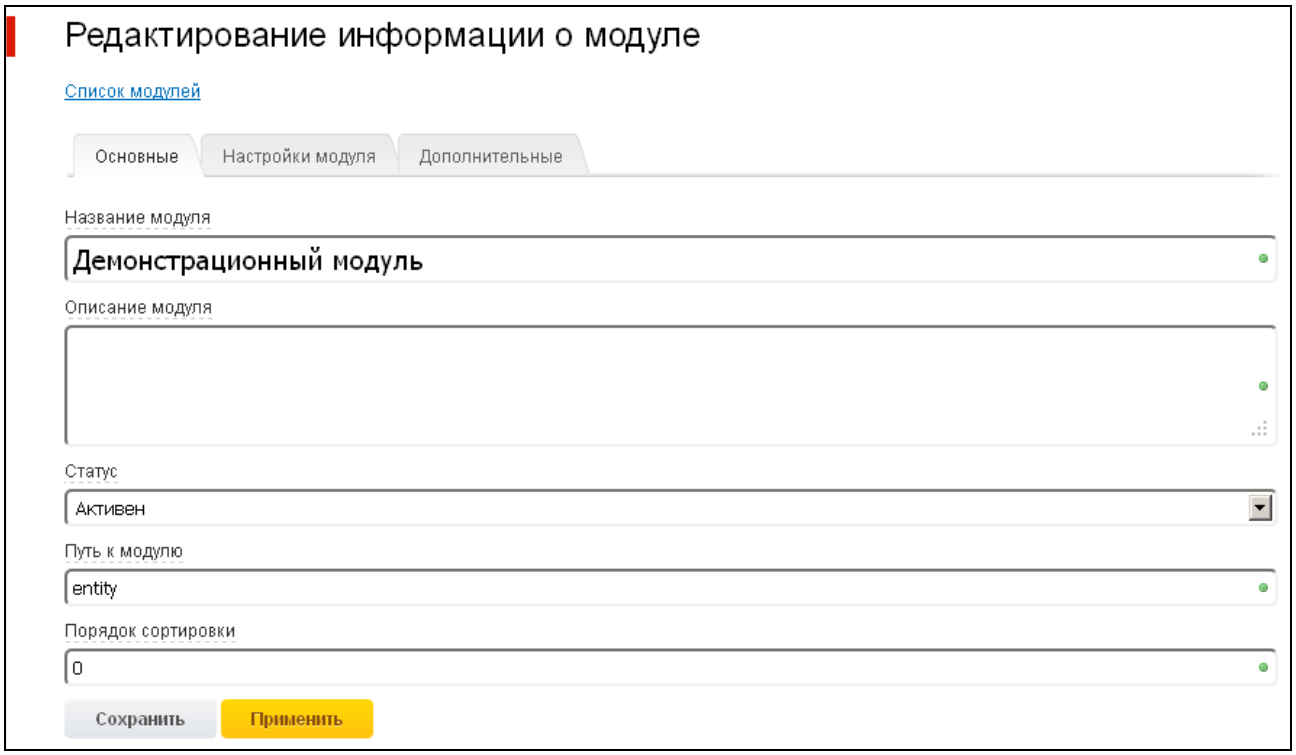
Метод `searchCallback()` используется для добавления к XML результата поиска информации дополнительных данных.

```
	/**
	 * Search callback function
	 * @param Search_Page_Model $oSearch_Page
	 * @return self
	 */
	public function searchCallback($oSearch_Page)
	{
		if ($oSearch_Page->module_value_id)
		{
			switch ($oSearch_Page->module_value_type)
			{
				case 1: // группы
					$oEntity_Group = Core_Entity::factory('Entity_Group')-
>find($oSearch_Page->module_value_id);

					!is_null($oEntity_Group->id) && $oSearch_Page-
>addEntity($oEntity_Group);
				break;
				case 2: // элементы
					$oEntity = Core_Entity::factory('Entity')-
>find($oSearch_Page->module_value_id);

					if (!is_null($oEntity->id))
					{
						$oEntity
							->showXmlComments(TRUE)
							->showXmlProperties(TRUE);

						$oSearch_Page
							->addEntity($oEntity)
							->addEntity($oEntity->Entity_Group);
					}
				break;
			}
		}

		return $this;
	}
```

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.

Работа с массивами

Класс `Core_Array` предназначен для безопасной работы с массивами. Для доступа к суперглобальным массивам `$_GET`, `$_POST` и `$_REQUEST` используйте, соответственно, методы `getGet($keyName)`, `getPost($keyName)`, `getRequest($keyName)`. Если значения не существовало, будет возвращено значение NULL, либо значение, переданное вторым аргументом, например:

```
// Возвращает значение параметра 'foo' из $_GET или NULL если ключ не существует
$value = Core_Array::getGet('foo');
```

```
// Возвращает значение параметра 'foo' из $_GET или 'bar' если ключ не существует
$value = Core_Array::getGet('foo', 'bar');
```