



UMI.CMS
СЕРДЦЕ ВАШЕГО САЙТА™

UMI.CMS

QuickStart

Guide

for developer.

Grigoriy Dobryakov

Готовим рабочее окружение

- 1.** Установите систему UMI.CMS без демо-сайта (для этого выберите соответствующий пункт в процессе установки).
- 2.** Зайдите в админку по адресу <http://www.ваш-домен.ru/admin/> указав логин и пароль администратора, заданный в процессе установки.
- 3.** Поместите пустой файл `my.xsl` в каталог `xsltTpls`.
- 4.** В модуле “Структура” в настройках модуля укажите этот файл в качестве шаблона по умолчанию: название – любое, имя файла – `my.xsl`, основной – отметить, и сохранить.
- 5.** Вернитесь в модуль “Структура”, наведите курсор на строку с доменом (в середине экрана) – появится плюсик. Нажмите на него и создайте первую страницу сайта.
- 6.** Создайте ещё несколько страниц и подстраниц, организовав соответствующую структуру (меню) будущего сайта. Рекомендуется ставить галочки “отображать в меню”.

Каталоги и файлы

Расположение	Назначение
classes/modules	Модули системы.
xsltTpls	Шаблоны дизайна в формате XSLT.
tpls	Шаблоны дизайна в формате TPL (устаревший).
xsl	Схемы XSL-преобразований и системные XSLT-шаблоны.
man	Тексты справки, которая выводится в админке в правой части.
files	Файлы, загружаемые пользователем.
libs	Библиотеки ядра.
smu	Файлы механизма обновлений.
sys-temp	Каталог для временных файлов системы.
.htaccess	Стандартный файл веб-сервера Apache, использующийся UMI.CMS для указания правил обработки запросов (mod_rewrite).
config.ini	Конфигурационный файл системы.

Если вы устанавливали систему с демо-сайтом, в каталогах tpls и xsltTpls уже есть готовые шаблоны. Вы можете использовать их для примеров.

Структура базы данных

В отличие от других систем, в UMI.CMS не создаётся отдельная SQL-таблица на каждый тип данных. Все данные хранятся в единой модели данных, состоящей из четырёх таблиц: объекты, типы данных, свойства (поля), и их значения.

Таблица	Столбец	Назначение
Объекты		
cms3_objects	id	Идентификатор объекта
	name	Название объекта
	type_id	Идентификатор типа
Значения свойств (полей) объектов		
cms3_objects_content	obj_id	Идентификатор объекта
	field_id	Идентификатор свойства (поля)
	*_val	Значение свойства
Типы данных		
cms3_objects_types	id	Идентификатор типа
	name	Название типа
	parent_id	Идентификатор родительского типа
Свойства (поля)		
cms3_objects_fields	id	Идентификатор свойства (поля)
	name	Название свойства

Разработчику сайта на UMI.CMS обычно не требуется работать напрямую с БД сайта. Начинающим разработчикам мы рекомендуем это делать только ради экспериментов для понимания её устройства. В настоящем проекте изменять БД напрямую категорически не рекомендуется - для этого есть средства API UMI.CMS.

Модель данных

Все данные в системе хранятся в виде объектов (object).

Каждый объект имеет определённый тип (type).

Каждый тип имеет определённый набор свойств (fields), описывающих характеристики объектов этого типа.

Набор свойств (fields) конкретного объекта описывается типом (type), которому он принадлежит.

Все значения свойств объектов физически хранятся в таблице `cms3_objects_content`, в строках с соответствующими `obj_id` и `field_id`.

Например, если у вас есть объект “апельсин” (`obj_id=28`), он относится к типу данных “фрукты” (`type_id=53`). У этого типа данных есть свойство “цвет” (`field_id=46`). Апельсин жёлтый, поэтому представьте что в таблице `cms3_objects_content` есть строка:

<code>obj_id</code>	<code>field_id</code>	Value
28	46	Жёлтый

На самом деле в БД UMI.CMS не один столбец Value, а несколько – для более корректного хранения значений разных типов (строки, числа, и так далее). Здесь мы просто приводим самый доступный пример.

Модули и методы

Для того, чтобы получить у системы какие-либо данные (например для вывода в шаблоне), в UMI.CMS используются модули и методы.

Каждый модуль находится в соответствующем каталоге в `/classes/modules/`. Основной файл модуля – `class.php`. Все публичные (`public`) методы этого класса – методы модуля.

В UMI.CMS принято писать модуль и метод через слэш, например:

Расположение	Назначение
<code>content/menu</code>	Основное меню сайта
<code>news/lastlist</code>	Список последних новостей
<code>catalog/getCategoryList</code>	Список разделов (категорий) каталога

Мы рекомендуем изучать `class.php`, но просим вас не менять эти файлы (и находящиеся рядом с ними тоже). При обновлении системы все системные файлы будут перезаписаны. Для ваших нужд при разработке кастомного функционала предназначены файлы с названиями вида `__custom.php`, которые не перезаписываются при обновлении.

Следующие примеры удобно рассматривать, когда в структуре сайта создано несколько страниц первого уровня, создана “лента” новостей и добавлено несколько новостей.

Запросы к модулю и методу

Чтобы увидеть, какие данные возвращает конкретный модуль и метод, введите в строке браузера запрос к udata, например:

<http://www.ваш-домен.ru/udata://content/menu>

<http://www.ваш-домен.ru/udata://news/lastlist/3/>

Важно: если вы не видите результата, откройте исходный код страницы, либо попробуйте использовать формат запроса без второго двоеточия и слэша:

<http://www.ваш-домен.ru/udata/content/menu/>

<http://www.ваш-домен.ru/udata/news/lastlist/3/>

Такое иногда бывает на локальном компьютере с установленным “Денвером”, например если вы установили “UMI.CMS Localpack для ознакомления под Windows”.

Первый параметр после “udata://” – это название модуля. Второй параметр – название метода. Остальные параметры (они перечисляются через слэш) являются необязательными. Узнать об их предназначениях можно двумя путями:

а) Посмотреть описание на <http://help-dev.umi-cms.ru/>

б) Посмотреть параметры функции в php-коде class.php этого модуля – они перечислены в определении этой функции (путь для истинных джедаев).

В данном примере “3” – это идентификатор “ленты” новостей в системе UMI.CMS. У вас это число может быть другим. Вы всегда можете его увидеть, перейдя в админке в редактирование нужного объекта (в данном случае в редактирование ленты новостей).

В следующих примерах часто используются идентификаторы объектов, страниц, категорий, - поэтому мы рекомендуем научиться смотреть их в админке прямо сейчас.

Структура udata

Запрос к модулю и методу всегда возвращает данные в формате XML. Их структура стандартизирована и единообразна для всех модулей системы.

Рассмотрим XML-данные на примере вышеприведённого запроса к модулю news и методу lastlist:

```
<udata module="news" method="lastlist">
```

- содержит наименование модуля и метода.

```
<items>...</items>
```

- содержит объекты, полученные в результате запроса.

```
<item id="4" link="/novosti/zagolovok/" xlink:href="upage://4" publish_time="1309266360"
lent_id="3" lent_name="Новости" lent_link="/novosti/">Zagolovok</item>
```

Расположение	Назначение
id	Идентификатор объекта в системе (в данном примере /admin/news/edit/4/)
link	Путь к странице (можно использовать для вывода в меню)
xlink:href	Подсказка метода получения объекта
publish_time	Время публикации в формате unixtime
lent_id	Идентификатор ленты новостей (в данном примере (/admin/news/edit/3/)
lent_name	Название ленты новостей (можно использовать для вывода в шаблоне)
lent_link	Путь к странице (можно использовать для вывода в меню)

Помните, что все элементы рассмотрены в <http://help-dev.umi-cms.ru/>

В стандарте XSLT и в документации UMI.CMS принято указывать вложенность элементов через слэш. Например, если у нас есть корневой элемент “`udata`”, в него вложен “`items`”, в который вложено несколько элементов “`item`”, то когда подразумевают эти элементы – пишут “`udata/items/item`”.

Такой формат записи похож на указание пути в файловой системе. Вы можете представлять себе структуру XML-данных как “дерево” или как вложенный массив, как вам удобнее.

Механизм, о котором идёт речь, называется xPath. Вам не нужно знать его в точности, чтобы понимать работу системы. Но тем не менее, мы считаем что вам это будет интересно. В том числе, есть прекрасный плагин для Mozilla Firefox <https://addons.mozilla.org/ru/firefox/addon/firepath/>

Получение одного объекта

Для работы следующих примеров вам потребуется включить в файле config.ini:

```
uobject.http.allow = "1"
```

```
upage.http.allow = "1"
```

Введите в строке браузера:

<http://www.ваш-сайт.ru/upage://4>

для получения данных о странице с идентификатором "4".

<http://www.ваш-сайт.ru/uobject://9>

для получения данных об объекте с идентификатором "9".

Если вы затрудняетесь определить, запрашивать ли страницу или объект, ориентируйтесь на параметр "xlink:href", рассмотренный в предыдущем примере.

Ответ на запрос об объекте содержит важную информацию:

```
<object id="9" type-id="8" ... >
```

type_id – это идентификатор типа объекта. Зная его, можно перейти к редактированию этого типа данных в админке по прямой ссылке:

http://www.ваш-домен.ru/admin/data/type_edit/8/

В админке вы увидите все свойства (поля) объектов данного типа, и сможете их отредактировать.

Вы можете использовать сокращённый синтаксис для получения значения только одного свойства (поля) объекта, например:

<http://www.ваш-сайт.ru/uobject://9.e-mail>

здесь если объект “9” – пользователь и имеет свойство “e-mail”, то вам вернётся XML с его значением:

```
<udata><property id="33" name="e-mail" type="string"><title>E-mail</title><value>user@domain.ru</value></property></udata>
```

Расположение	Назначение
id	Идентификатор свойства (поля) в системе (в данном примере /admin/news/edit/4/)
name	Системное название свойства
type	Тип свойства
title	Видимое название свойства

Данные о текущей странице

Для получения данных о текущей странице введите в строке браузера, например:

<http://www.ваш-сайт.ru/.xml>

<http://www.ваш-сайт.ru/novosti/.xml>

В XSLT-шаблонизаторе именно эти данные доступны в элементе result. Об этом будет рассказано в следующих главах.

Интеграция HTML-шаблона

Для работы со следующими примерами рекомендуется включить режим отладки XSLT в файле config.ini:

```
[debug]
enabled="1"
```

Подготовим начальный шаблон в файле xsltTrpls/my.xsl, который мы создали в первой главе:

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:xlink="http://www.w3.org/TR/xlink">
<xsl:output encoding="utf-8" method="html" indent="yes"/>

<xsl:template match="/">
<html>
<head>
<title></title>
</head>
<body>
</body>
</html>
</xsl:template>

</xsl:stylesheet>
```

Структура шаблона:

Тэг	Назначение
xml version...	Заглавный тэг любого XML-документа
xsl:stylesheet	Заглавный тэг XSL-шаблона
xsl:output	Указание режимов вывода
xsl:template	Непосредственно код шаблона

В этом примере код, написанный чёрным цветом, является стандартным для всех шаблонов. Используйте его во всех своих шаблонах как удобную заготовку. Не забывайте закрывать все тэги.

Директива `match="/"` здесь говорит о том, что этот шаблон является глобальным. Такой шаблон обязательно должен быть, и он должен быть только один.

Вывод простых значений

Напомним, что все данные в системе выводятся в формате XML, а ваша задача – оформить их вывод в шаблоне. Рекомендуем всегда перед тем, как выводить что-либо в шаблоне, открывать в отдельном окне браузера соответствующую XML-выдачу:

<http://www.ваш-домен.ru/.xml>

Допустим, нам нужно вывести html title (заголовок страницы). Найдите его в структуре XML-данных. Это значение элемента `/result/page/name`. Для его вывода в шаблоне используйте:

```
<title><xsl:value-of select="/result/page/name" /></title>
```

Значение тэга `h1` вывести немногим сложнее. Взгляните на ту же структуру XML и найдите свойство `h1`. Это значение в блоке `/result/page/properties` с названием `name="h1"`. Поскольку свойств (`properties`) может быть много, для вывода нужно уточнить условие выборки:

```
<h1>  
<xsl:value-of  
select="/result/page/properties/group[@name='common']/property[@name='h1']/value"/>  
</h1>
```

Здесь анализируется элемент `properties`, из которого извлекается группа (`group`) с названием `name='common'`. Далее из этой группы извлекается свойство (`property`) с названием `name='h1'`. Затем у этого свойства берётся значение (`value`). Оно и выводится в шаблон.

Вывод контента выполняется аналогично:

```
<xsl:value-of select="/result/page/properties/group/property[@name='content']/value"  
disable-output-escaping="yes" />
```

Подключение подшаблонов

Рассмотрим вывод списка новостей в отдельном шаблоне, который мы разместим в отдельном файле. Назовите его, например, news-column.xsl и поместите рядом с my.xsl. Для его подключения используйте директиву:

```
<xsl:include href="news-column.xsl"/>
```

Не оставляйте этот файл пустым! Вставьте в этот файл заготовку, рассмотренную в конце главы “Интеграция HTML-шаблона”.

Важно: директива xsl:include не должна быть внутри xsl:template. Размещайте её выше него.

На примере основного меню и списка новостей в следующей главе рассмотрим вывод любых списков в системе. Помните, что списки любых других данных выводятся в точности таким же образом.

Вывод списков

Чтобы вывести более сложные данные, чем просто свойства страницы, в первую очередь найдите подходящие модуль и метод. Затем запросите в браузере соответствующую XML-выдачу и внимательно на неё посмотрите.

В первом примере мы выводим меню основных разделов сайта:

<http://www.ваш-сайт.ru/udata://content/menu>

Для работы с этим примером не забудьте включить галочки “Отображать в меню” в редактировании страниц сайта (в админке).

Интересующие нас пункты меню находятся в элементе `/udata/items/`. Каждый пункт меню – это отдельный элемент `item`. В шаблоне страницы определите место для меню, и вставьте туда этот запрос в таком формате:

```
<div id="menu">
<ul>
<xsl:apply-templates select="document('udata://content/menu')/udata/items" />
</ul>
</div>
```

Обратите внимание на прямое сходство этого запроса с тем, который вы видите в браузере. Эта директива определяет, что в обработку следует передать именно эти указанные XML-данные.

В шаблоне следует указать инструкцию - условие, по которому он применится к подходящим XML-данным. Добавьте новый блок вне существующих блоков “`xsl:template`”:

```
<xsl:template match="udata[@module='content' and @method='menu']/items/item">
<li>
<a href="{@link}"><xsl:value-of select="." /></a>
```

```
</li>
</xsl:template>
```

Здесь директива “match” указывает, что данный шаблон должен быть применён к элементу udata (если в нём module='content' и method='menu') и вложенным в него элементам items/item. Если точнее - шаблон будет применён конкретно к каждому item-у.

Так как этих элементов несколько, шаблон применится к каждому из них по-порядку. Таким образом, на страницу будет выведен их список, как если бы это сделано циклом в обычном языке программирования.

Управляя html-тэгами, вы можете оформить это меню как списком (как показано в примере), так и строкой, таблицей, блоками или любым другим путём.

Тэг	Назначение
xsl:template	Заглавный тэг подшаблона
li	Обычный html-элемент списка (здесь нужен только для оформления)
xsl:value-of	Вывод значения элемента или свойства
“.”	Значение текущего элемента (в данном случае – название страницы)

Во втором примере мы выводим список новостей:

<http://www.ваш-сайт.ru/udata://news/lastlist/3>

(Если вы забыли, что это такое, вернитесь в главу “Запросы к модулю и методу”)

Интересующие нас новости находятся в элементе `/udata/items/`.

Каждый анонс новости – это отдельный элемент `item`.

Определите в главном шаблоне (`my.xsl`) место для вывода списка новостей, и вставьте туда этот запрос в таком формате:

```
<ul id="news_block">
<xsl:apply-templates select="document('udata://news/lastlist/3/)/udata/items" />
</ul>
```

В шаблоне следует указать инструкцию - условие, по которому он применится к подходящим XML-данным:

```
<xsl:template match="udata[@module='news' and @method='lastlist']/items/item">
<!-- здесь будет html-код шаблона -->
</xsl:template>
```

Отредактируйте `news-column.xsl` и задайте оформление элемента, чтобы новости выводились в список с датами публикации и заголовками:

```
<xsl:template match="udata[@module='news' and @method='lastlist']/items/item">
<li>
<xsl:value-of select="@publish_time"/>
&#160;
<xsl:value-of select="."/>
</li>
</xsl:template>
```

Структура шаблона:

Тэг	Назначение
xsl:template	Заглавный тэг подшаблона
li	Обычный html-элемент списка (здесь нужен только для оформления)
xsl:value-of	Вывод значения элемента или свойства
 	Entity-аналог неразрывного пробела
@publish_time	Свойство новости – дата/время публикации
“.”	Значение текущего элемента (в данном случае - заголовок новости)

В результате на странице сайта вы должны увидеть заголовок h1 и список новостей. Не удивляйтесь, если даты новостей будут выглядеть не как даты, а как большие числа – их преобразование в “человекочитаемый” формат мы рассмотрим ниже.

В чём здесь магия?

Здесь кроется один из важных моментов изучения XSLT-шаблонизатора. В нём вы не найдёте привычных “включений” шаблонов простыми `<? require ?>` или `<? include ?>`.

Все шаблоны подключаются всегда. Директивой `select` вы просто указываете, что некоторый XML-запрос требуется отправить в обработку, а результат вставить в данное место шаблона. В этом месте вы не управляете тем, какой шаблон будет выбран.

Все директивы `select` отправляют в обработку множество XML-данных вместе. Выбор подходящего шаблона осуществляется по условию, указанному в тэге `match` самого шаблона. Шаблон применяется к подходящим по условию XML-данным.

XSLT-шаблонизатор всегда пытается применить все шаблоны (templates) ко всем тэгам `select`. Ваша задача как разработчика – указать все необходимые `select`, а затем правильно определить `match` для каждого шаблона.

Вы можете расположить шаблоны в отдельных файлах и подкаталогах так, как считаете удобным (в рамках каталога `xsltTpls`). Когда система приступает к обработке и выводу шаблона, она всё равно их “внутри себя” склеивает в один большой файл.

Вывод типовых страниц

Практически любой сайт имеет типовые страницы: страница со списком новостей, страница со списком товаров и т.д.

В предыдущих главах мы рассмотрели блок со списком новостей, который можно вывести (например) в правом столбце сайта на всех страницах. Теперь рассмотрим как сделать, чтобы на странице новостей были новости, а на странице каталога – товары каталога.

Для того чтобы было удобно рассматривать следующий пример, создайте в админке раздел каталога и несколько товаров.

Откройте в браузере XML со списком товаров:

<http://www.ваш-сайт.ru/udata://catalog/getObjectsList//7>

(Если вы забыли, что это такое и почему здесь стоит цифра “7”, вернитесь в главу “Запросы к модулю и методу”)

Список товаров вложен в элемент `lines`, каждый товар – это отдельный `item`. Создайте новый файл, назвав его например `catalog-column.xml`, поместите в него вашу типовую заготовку `xsl`-шаблона и подключите в основном шаблоне (`xsl:include`).

Мы предположим, что список товаров будет выводиться в блоке основного контента страницы. Задайте в основном шаблоне в блоке, предназначенном для вывода контента, следующую директиву:

```
<div id="content">
...
<xsl:apply-templates select="result" />
</div>
```

Эта директива указывает, что в обработку надо передать XML-данные о текущей странице.

“result” - это то, что вы видите в браузере, когда дописываете “.xml” к любой странице на UMI.CMS. Откройте в браузере вашу страницу, на которой должен быть каталог товаров, допишите “.xml” и обратите внимание на module и method в XML-выдаче. Если вы всё сделали правильно, вы увидите модуль catalog и метод category.

Далее мы напишем шаблон, предназначенный для вывода списка товаров на странице каталога. В тэге match мы укажем, что речь идёт о странице каталога (module='catalog' и method='category'). В свою очередь, метод выводящий список товаров называется getObjectList. Ему надо передать id категории каталога:

```
<xsl:template match="result[@module='catalog' and @method='category']">
<h2>Список товаров</h2>
<xsl:apply-templates
select="document(concat('udata://catalog/getObjectsList/',page/@id,''))/udata" />
</xsl:template>
```

Почему здесь два слэша после getObjectList? Обратитесь к документации на этот метод, и вы увидите, что первым параметром следует название шаблона. Если бы этот параметр был указан, он был бы именно между этими двумя слэшами. Но мы его не указываем, то есть оставляем пустым. Атрибут @id – это id текущей категории каталога, который вы могли увидеть в XML-выдаче, обсуждаемой на предыдущей странице.

Затем в файле catalog-column.xsl укажите шаблон, который применится к этим XML-данным:

```
<xsl:template match="udata[@module='catalog' and @method='getObjectList']">
<ul>
<xsl:apply-templates select="lines/item"/>
</ul>
</xsl:template>
```

Этот шаблон направляет в обработку каждый элемент item внутри lines. Напишем шаблон и для него:

```
<a href="{@link}"><xsl:value-of select="."/;></a>
```

Если вы всё сделали правильно, то у вас на странице каталога выведется список товаров. Их названия будут ссылками на страницы каждого товара.

Как вывести то, чего нет в XML

Для вывода информации, которой (на первый взгляд) просто нет в готовых XML-данных UMI.CMS, есть два пути:

1. Посмотреть в <http://help-dev.umi-cms.ru/> и найти запрос, возвращающий нужные XML-данные.
2. Просто написать метод, который будет их возвращать.

Создание метода выходит за рамки этого руководства и подробно описано в http://api.umi-cms.ru/module_dev.createModule.html. В следующем примере мы будем считать, что у вас уже есть подходящий модуль и метод.

Рассмотрим пример преобразования timestamp даты публикации новости в “человекочитаемый” формат даты. Откройте в браузере:

[http://www.ваш-сайт.ru/udata://system/convertDate/1239933231/\(d.m.Y\)](http://www.ваш-сайт.ru/udata://system/convertDate/1239933231/(d.m.Y))

Здесь мы обращаемся к модулю system, методу convertDate, передавая ему параметры 1239933231 и “d.m.Y”.

Чтобы вам было легче понять, представьте аналог в PHP:

```
function convertDate($timestamp,$format) { ... }  
echo ( convertDate(1239933231,"d.m.Y") );
```

Здесь в запросе d.m.Y указано в круглых скобках, чтобы точно указать что эту строку нужно считать одним параметром.

Теперь в шаблоне news-column.xsl в том месте, где у вас выводился @publish_time, нужно заменить его вывод на применение шаблона:

```
<xsl:apply-templates select="@publish_time"/>
```

Затем написать сам шаблон (помните, что он может быть в любом месте, поэтому просто располагайте его там где вам удобно – но вне других шаблонов):

```
<xsl:template match="@publish_time">
<xsl:value-of select="document(concat('udata://system/convertDate/',.,'/d.m.Y'))/udata" />
</xsl:template>
```

Структура шаблона:

Тэг	Назначение
xsl:template	Заглавный тэг подшаблона с условием применения к атрибуту publish_time
concat	Операция составления строк (конкатенации), то же самое что и точка для “склеивания” частей строк в PHP
xsl:value-of	Вывод значения того XML-узла, который будет получен в результате работы этого запроса
“.”	Значение текущего элемента (в данном случае – числовое значение параметра publish_time)

Зачем здесь concat и точка посередине? Точка в XSLT – значение текущего элемента. Текущий элемент – publish_time, как указано в match. Функция concat “склеивает” три подстроки в одну целую: запрос к udata, текущее значение элемента (точка), и формат вывода даты “d.m.Y” – день, месяц, год.

Почему всё именно так?

Почему бы просто не вставлять в шаблоны php-код, как вы привыкли раньше? Вот причины, о которых мы говорим в первую очередь:

- 1.** XSLT – стандарт разработки, используемый Google и Yandex. Применяя его, вы вступаете в мировое сообщество профессиональных разработчиков и повышаете свой скилл.
- 2.** MVC и разделение труда. Если вы читали о паттерне MVC, то вы знаете преимущества размещения бизнес-логики в ядре системы (model и controller), а логики представления – в шаблонах (view).
- 3.** Система ведёт себя так, как вы ожидаете. Формат хранения данных и формат их вывода – известны и всегда предсказуемы.
- 4.** Быстрое внедрение. Достаточно один раз подготовить набор готовых шаблонов, чтобы быстро предоставлять каждому новому заказчику возможность работать с сайтом – ещё до интеграции дизайна.
- 5.** Неважно, как долго клиент будет согласовывать дизайн и структуру сайта – при помощи наших технологий вы просто переставляете блоки apply-templates и меняете стили в css. При этом вам не нужно менять ни вывод данных, ни сами подшаблоны.

Рекомендации разработчику

- 1.** Всегда создавайте свои шаблоны, файлы, типы данных. Помните, что почти все системные элементы будут переписаны дефолтными значениями при обновлении системы.
- 2.** Помните, что нельзя включать подшаблон в шаблон. Они не должны пересекаться.
- 3.** Перед тем, как вывести какие-либо данные на странице, всегда ищите подходящий модуль и метод, смотрите какие параметры он готов принять, и какие XML-данные отдаёт.
- 4.** Не забывайте включать режим отладки (debug) и доступ к протоколу uobject в config.ini при разработке сайта, но отключать на “боевом” сайте.
- 5.** Называйте каталоги и файлы подшаблонов понятными именами.
- 6.** В условиях (match) привыкайте использовать name вместо id, так как id может быть разным на разных сайтах.

Что дальше?

UMI.CMS Dev Book

<http://www.umi-cms.ru/support/docs/xslt-umi-devbook/> (PDF)

Документация по модулям и методам

<http://help-dev.umi-cms.ru/>

UMI.Wiki

<http://wiki.umisoft.ru/>

API UMI.CMS

<http://api.umi-cms.ru/>

Заключение

Об авторе:

Григорий Добряков, технический директор Юмисофт.

<http://blog.umi-cms.ru/dobryakov/>

mailto: dg@umisoft.ru

Благодарности:

Станислав Голензовский, руководитель Службы Заботы.

Виктор Фёдоров, ведущий XSLT-верстальщик.

Виталий Шубин, ведущий специалист Службы Заботы.