Углубленный курс по XSLT-шаблонизатору

Даниил Сироткин
UmiHelp.ru

Подключение скриптов для работы с корзиной без перезагрузки

1. Подключаем скрипт /js/site/__common.js

Скрипт должен появляться после вызова макроса
%system includeQuickEditJs()%

```
<head>
    ...
    <xsl:value-of select="document('udata://system/includeQuickEditJs')/udata" disable-output-escaping="yes" />
    ...
    <script type='text/javascript' src='/js/site/__common.js'></script>
    ...
</head>
```

2. Прописываем необходимые классы в верстку корзины для взаимодействия со скриптом:

.cart_summary — контейнер для суммарной стоимости товаров в корзине
.cart_item_[element_id] — контейнер для товара
.cart_item_price_[element_id] — контейнер для итоговой цены конкретного товара
.basket_info_summary — контейнер для итоговой информации по корзине (общее кол-во и цена)

Подключение скриптов для работы с корзиной без перезагрузки

2.1 Пример шаблона для краткой корзины в шапке сайта:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE xsl:stylesheet SYSTEM "ulang://i18n/constants.dtd:file">

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="udata[@module = 'emarket' and @method = 'cart']" mode="basket" >
        <div class="short_basket top_box light_blue_box shbasket">
            <div class="short_basket_in">
                <h2><a href="{/result/@pre-lang}emarket/cart/">Ваша корзина</a></h2>
                <span class="basket_info_summary">товаров: пока нет <br/>
                общая стоимость: -
                </span>
            </div>
        </div>
    </xsl:template>

    <xsl:template match="udata[@module = 'emarket' and @method = 'cart' and count(items/item) &gt; 0]" mode="basket">
        <div class="short_basket top_box light_blue_box shbasket">
            <div class="short_basket_in">
                <h2><a href="{/result/@pre-lang}emarket/cart/">Ваша корзина</a></h2>
                <xsl:apply-templates select="summary" mode="basket" />

            </div>
        </div>

    </xsl:template>

    <xsl:template match="summary" mode="basket">
        <xsl:text>Корзина пуста</xsl:text>
    </xsl:template>

    <xsl:template match="summary[amount &gt; 0]" mode="basket">
        <span class="basket_info_summary">товаров: <xsl:apply-templates select="amount" /> <br/>
        общая стоимость: <xsl:value-of select="price" /> руб
        </span>
    </xsl:template>
</xsl:stylesheet>
```

Подключение скриптов для работы с корзиной без перезагрузки

2.2 Некоторые особенности шаблона для страницы корзины:
- Всю корзину (шаблон **<xsl:template match="udata[@method = 'cart'][count(items/item) > 0]">**) необходимо помещать в контейнер с классом «**basket**»

- Ссылку на удаление товара из корзины (в шаблоне **<xsl:template match="udata[@method = 'cart']//item">**) необходимо дополнить js функцией удаления, для удаления товара без перезагрузки необходимо добавить класс (**<a href="{$lang-prefix}/emarket/basket/remove/item/{@id}/" id="{@id}" class="del" />**)или явно прописать вызов функции при нажатии на ссылку (**(X)**)

- Контейнер для товара (в шаблоне **<xsl:template match="udata[@method = 'cart']//item">**) необходимо пометить классом **cart_item_{@id}**.
Пример: **<tr class="cart_item_{@id}">**

2.3 Некоторые особенности ссылки на добавление товара в корзину.
Чтобы избежать сложностей с id, вместо **<xsl:apply-templates select="document(concat('udata://emarket/basketAddLink/', @id))/udata" />** используйте ссылку вида:
в корзину

id **"add_basket_{@id}"** и class **"basket_list"** обязательны для связки со скриптом **/js/site/__common.js**

Подключение скриптов для работы с корзиной без перезагрузки

3. Отладка.

Улучшить скрипт взаимодействия можно путем изменения файла /js/site/basket.js, который подключается файлом /js/site/__common.js .
Для просмотра объекта, который система возвращает при добавлении, удалении или изменении товаров в корзине, запустите в адресной строке вызов данных функций и вы увидите весь объект, в том числе топологию переменных
http://localhost/udata/emarket/basket/put/element/64.json (64 это id элемента в структуре)
http://localhost/udata//emarket/basket/remove/item/672.json (672 это id данного объекта)

Для контроля работы скрипта используйте дополнение firebug. Вкладка «Консоль» покажет ошибки в скрипте.
На время отладки файлы ('/js/client/utilities.js', '/js/client/basket.js', '/js/site/basket.js', '/js/site/forms.js', '/js/site/message.js', '/js/site/captcha.js', '/js/jquery/jquery.cookie.js'), подключаемые файлом /js/site/__common.js, подключите в head шаблона сразу под вызовом /js/site/__common.js. Не забудьте удалить их повторный вызов из файла /js/site/__common.js.
Это необходимо для того, чтобы видеть в консоли firebug ошибки работы скриптов.

Выборки из Базы Данных: протокол USel

1. Синтаксис и пример протокола

REST протокол USel позволяет делать выборки из базы используя шаблоны, которые представляют собой XML-файлы в определенном формате. Все XML-файлы, которые содержат шаблоны USel должны находится в папке /usels/.

Просмотреть результаты, возвращаемые по указанному шаблону, можно набрав в адресной строке http://ваш_сайт/usel/имя_шаблона (просмотр через http резкультатов протокола usel может быть запрещен в config.ini)

Пример шаблона для вывода 5 страниц, типа данных объект каталога, из раздела каталога с id, переданных первым параметром «{1}», имеющих галочку в поле с идентификатором «novinka», отсортированных по цене от большего к меньшему:

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <selection>
3      <target result="pages">
4          <type module="catalog" method="object" />
5          <category depth="1">{1}</category>
6      </target>
7      <property name="novinka" select="1" />
8      <sort order="descending">price</sort>
9      <limit page="0">5</limit>
10 </selection>
```

Документация по протоколу Usel доступна по адресу
http://help-dev.umi-cms.ru/chapter.XSLTTemplates.usel.html

Создание кастомных макросов

1. Создание макроса «hello world».

Все макросы, не привязанные к каким-то модулям, удобнее создавать в файле **/classes/modules/custom.php**. Так как макросы, написанные в этом файле, не требуют определения прав на них, то любой посетитель сайта увидит результаты работы данного макроса:

- разместим в файле **/classes/modules/custom.php** после надписи «**//TODO: Write your own macroses here»** наш макрос.

```
public function hello_world() {
    return 'hello world'; //вывести строчку «hello_world»
}
```

! Не забывайте проверять наличие файла в кодировке utf-8, особенно, если вы пишете в нем кириллицей.

- для вызова нашего макроса пропишем в нужном шаблоне строчку
<xsl:value-of select="document('udata://custom/hello_world')/udata" />

Создание кастомных макросов

2. Создание макроса для вывода названия страницы:

- модифицируем макрос, чтобы он выводил название страницы с заданным id

```
public function page_name($element_id=NULL) {
    if(!$element_id) return; //проверяем передан ли id страницы
    $element = umiHierarchy::getInstance()->getElement($element_id); // получаем
    umiHierarchyElement, либо false, если страница не существует
    if (!$element instanceof umiHierarchyElement) {
        throw new publicException(getLabel('error-page-does-not-exist'));
        // выбрасываем исключение
    }
    return $element->getName(); // выводим имя страницы
}
```

- для вызова данного макроса пропишем в нужном шаблоне строчку
<xsl:value-of select="document('udata://custom/page_name/64')/udata" />
64 - id_какой-нибудь_страницы

Создание кастомных макросов

3. Модификация макроса %catalog getCategoryList()% и назначение ему прав

- Создамим макрос **%catalog custom_getCategoryList()%** на основе **%catalog getCategoryList()%**. Отличием нашего макроса будет являться выделение текущего раздела.

- Так как наш модифицированный метод будет работать с элементами каталога, то логично разместить его в файле **/classes/modules/catalog/__custom.php** (такой файл есть в каждом модуле).

- Так как наш макрос рамещается не в **/classes/modules/custom.php**, необходимо прописать ему права на использование. Для этого мы создадим файл **permissions.custom.php** в этой же папке (**/classes/modules/**), который дополнит разделение прав прописанных в файле **permissions.php.** Итак, пропишем в файл наш макрос

```php
<?php
    $permissions = Array(
        'view' => Array('custom_getCategoryList')
    );
?>
```

- Теперь копируем макрос **%catalog getCategoryList()%** (его можно найти в файле **/classes/modules/class.php**) в файл **/classes/modules/catalog/__custom.php,** сразу после надписи **//TODO: Write here your own macroses**

- Вызываем макрос **<xsl:apply-templates select="document(concat('udata://catalog/custom_getCategoryList/void/', @id))/udata" />** , **@id** - это id страницы, внутри которой лежат наши подразделы.

Создание кастомных макросов

4. Правила вывода данных кастомным макросом

Вывод результатов кастомного скрипта в **xslt** шаблонизаторе - это вывод массива с результатами, который преобразуется в xml файл, поэтому важно правильно задавать название элемента массива (node, subnodes, void,attribute).

Пример xml, который сформиран в результате работы нашего кастомного скрипта **hello_world:**

```
<udata module="custom" method="hello_world" generation-time="0.001274">hello world</udata>
<!-- This page generated in 0.051584 secs -->
```

Пример xml, который сформиран в результате работы нашего кастомного скрипта **custom_getCategoryList**:

```
- <udata category-id="19" module="catalog" method="custom_getCategoryList" generation-time="0.031831">
  - <items>
      <item id="73" link="/market/1_podrazdel/" xlink:href="upage://73">1 подраздел</item>
      <item id="74" link="/market/2_podrazdel/" xlink:href="upage://74">2 подраздел</item>
    </items>
    <total>2</total>
    <per_page>25</per_page>
  </udata>
<!-- This page generated in 0.129058 secs -->
```

Протоколы выборки данных «UmiSelection» и «Selector»

1. Основы выборки.

UMI.CMS предлагается специальное api для работы с базой данных и организации выборки из нее.
Изначально в api umi был класс umiSelectionsParser. Этот объект имеет всего два статических метода: runSelection и runSelectionCounts, каждый из которых принимает единственный аргумент – объект класса umiSelection. Задача umiSelectionsParser – произвести выборку объектов или элементов иерархии в соответствии с критериями, хранимыми в объекте umiSelection. С более подробной информацией вы можете ознокомиться в официальной документации по api для umi-cms.

Спустя несколько релизов в umi-cms появился новый класс Selector, который был улучшен и оптимизирован относительно umiSelection.

Каким бы классом вы не пользовались, выборка дает вам возможность, не погружаясь в особенности базы данных, сделать запрос на выборку тех или иных объектов используя функционал api.

Протоколы выборки данных «UmiSelection» и «Selector»

2. Примеры использования и различия umiSelection и Selector.

Выборка всех новостей из новостной ленты с id=«21» средствами umiSelection + umiSelectionParser:

```
$hierarchy_type_id = umiHierarchyTypesCollection::getInstance()->getTypeByName("news", "item")->getId();
$iHierarchyTypeId = umiHierarchyTypesCollection::getInstance()->getTypeByName("banners", "banner")->getId();
$NewsItemTypeId =  umiObjectTypesCollection::getInstance()->getTypeByHierarchyTypeId($hierarchy_type_id);
$NewsItemType = umiObjectTypesCollection::getInstance()->getType($NewsItemTypeId);
$isVisibleFldId = $NewsItemType->getFieldId('is_unindexed');
$publishTimeFldId = $NewsItemType->getFieldId('publish_time');
$parent_id=21;
$per_page = 4;
$curr_page = 0;

$sel = new umiSelection; //получаем объект класса umiSelection
$sel->addElementType($hierarchy_type_id); //берем все новости
$sel->addHierarchyFilter($parent_id, 0); //добавляем ограничение по id страницы, внутри которой будем искать новости
$sel->addPropertyFilterEqual($isVisibleFldId, 1); //отбираем только те новости, у которых в поле "Исключить из поиска" стоит галочка
$sel->addPermissions(); //учитывает права текущего пользователя на данные объекты
$sel->setOrderByProperty($publishTimeFldId, 0);//отсортировать результат по полю "Дата публикации" от последнего к первому
$sel->addLimit($per_page, $curr_page); // per_page - кол-во элементов на страницу; $curr_page - номер текущей страницы

$result = umiSelectionsParser::runSelection($sel); //получаем результат выборки
$total = umiSelectionsParser::runSelectionCounts($sel); //получаем количество результатов выборки без учета ограничение на постраничный вывод указанный в addLimit
```

Протоколы выборки данных «UmiSelection» и «Selector»

2. Примеры использования и различия umiSelection и Selector.

Выборка всех новостей из новостной ленты с id=«21» средствами Selector:

```
$parent_id=21;
$per_page = 4;
$curr_page = 0;
$offset = $curr_page * $per_page;

$pages = new selector('pages');
$pages = new selector('pages');
$pages->types('object-type')->name('news', 'item');
$pages->where('hierarchy')->page($parent_id)->childs(1); //childs(1) - взять все элементы текущего уровня
$pages->where('is_unindexed')->equals(1); //отбираем только те новости, у которых в поле "Исключить из поиска" стоит галочка
$pages->order('publish_time')->desc(); //отсортировать результат по полю "Дата публикации" от последнего к первому
$pages->limit($offset, $per_page); //выводим элементы начиная с $offset в количестве $per_page
```

Использование событийной модели umiEvent

1. Описание событийной модели.

Событие (**event**) - это то, что происходит в результате некоторых действий пользователя или самой системы.

Событийная модель включает в себя следующие понятия:

- Точка вызова (**Event point**) - точка вызова события. Определяется в методе-инициаторе события.
- Перехватчик события (**Event listener**) - отслеживает определенное событие и запускает соответствующий обработчик.
- Обработчик события (**Event handler**) -метод, который исполняется по факту возникновения события.

Примеры использования событийной модели в UMI.CMS:

- При оформление заказа менеджеру высылается письмо о том, что совершен заказ.
- Рассылка подписки при обращении к cron.php
- Импорт подключенных RSS лент при обращении к cron.php
- Пересчет количества сообщений в топике форума при включение\выключении сообщения.

Использование событийной модели umiEvent

2. Обзор основных точек вызовов.

Точки вызова (umiEventPoint), которые можно использовать в административном режиме:
- **systemModifyElement -** изменение страницы;
- **systemCreateElement** - создание страницы;
- **systemSwitchElementActivit -** изменение активности страницы;
- **systemMoveElement -** перемещение страницы в структуре сайта;
- **systemDeleteElement -** удаление страницы в корзину;

- **systemModifyObject -** изменение объекта;
- **systemCreateObject -** создание объекта;
- **systemDeleteObject -** удаление объекта, без возможности восстановления.

Точки вызова, которые можно использовать на клиентской части сайта:
- **comments_message_post_do** - добавление комментария к какой-либо странице сайта (comments/class.php);
- **blogs20PostAdded** - добавление поста в блог;
- **blogs20CommentAdded** - добавление комментария к посту блога;
- **order-status-changed** - изменение статуса заказа (вызывает при оформлении заказа);
- **users_registrate** - регистрация нового пользователя;
- **webforms_post** - отправка сообщения из формы обратной связи;
- **umiObjectProperty_loadPriceValue** - загрузка значения поля типа Price.

Использование событийной модели umiEvent

3. Назначение обработчика события.

Создадим обработчик события, который при добавлении регистрации пользователя, будет добавлять его e-mail к списку рассылки.

3.1 Создание перехватчика события (**Event listener**):

Создадим файл **custom_events.php** в директории модуля dispatches (**/classes/modules/dispatches/**) и пропишем туда перехватчик события:

```php
<?php
    new umiEventListener('users_registrate', 'dispatches', 'autoSubscribe');
?>
```

Мы назначили обработчик события - некий метод autoSubscribe модуля dispatches, который мы опишем ниже.

Использование событийной модели umiEvent

3.2 Создание обработчика события (Event handler).

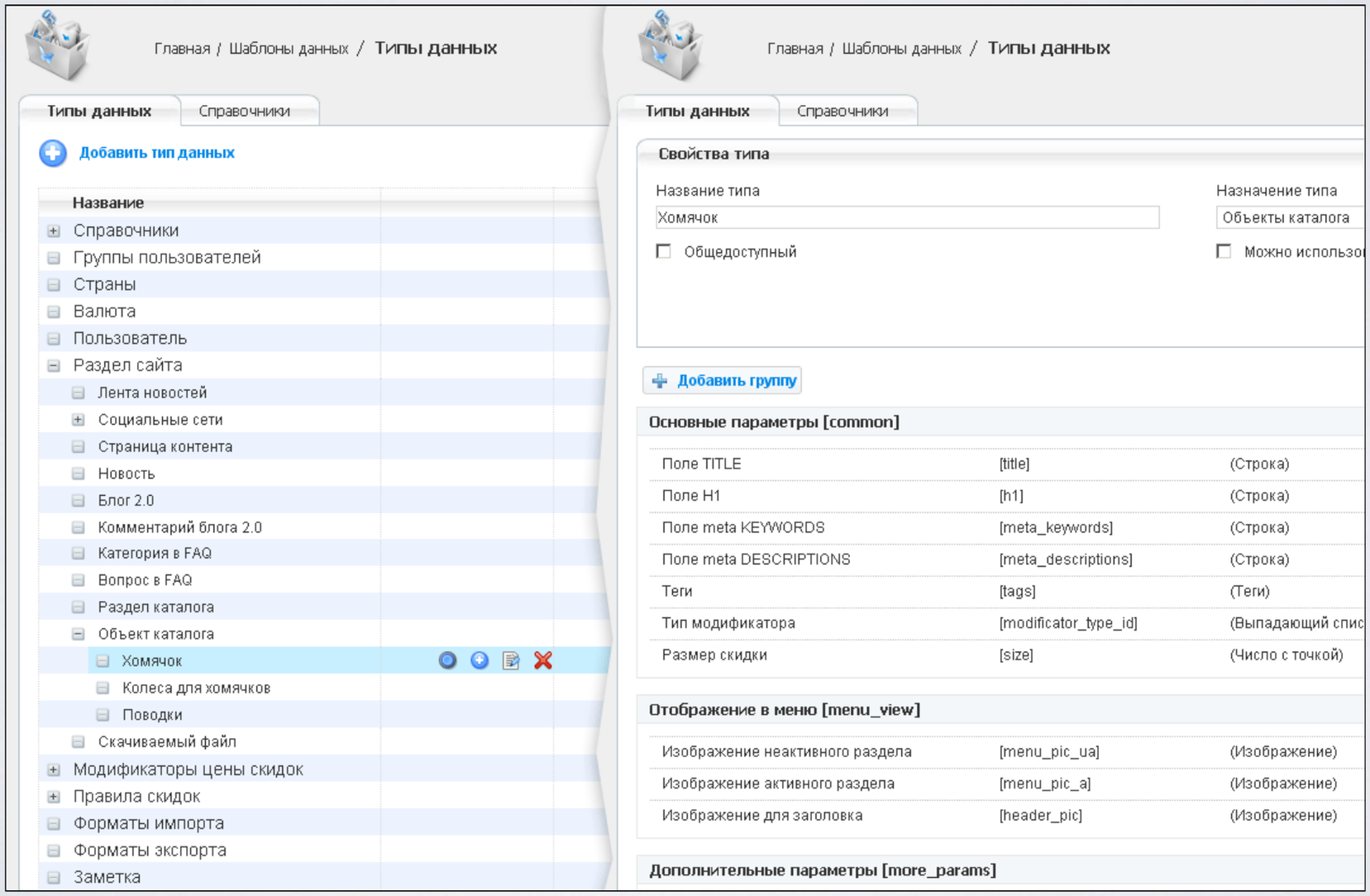
Наш обработчик разместим в файле **/classes/modules/dispatches/__custom.php**

```php
<?php
    abstract class __custom_dispatches {
        //TODO: Write here your own macroses

        public function autoSubscribe(umiEventPoint $event) {
            if ($event->getMode() == "after") {
                $user_id  = $event->getParam('user_id');
                $objects = umiObjectsCollection::getInstance();
                $user = $objects->getObject($user_id);
                $email = $user->getValue('e-mail');
                $fname = $user->fname;
                $lname = $user->lname;
                if(getRequest("autoSubscribe")) {
                    return $subscriber_id = $this->import_subscriber($email, $fname, $lname);
                } else {
                    return false;
                }
            }
        }

    };
?>
```

Различие страниц и объектов при разработке макросов

В UMI.CMS все данные (баннер, страница контента, пользователь и т.д.) хранятся в виде объектов и никаких сущностей больше нет.

Наборы характеристик объектов определяются «полями», которые группируются в «группы полей», что наглядно видно в модуле «Шаблоны данных»



Различие страниц и объектов при разработке макросов

Страницы в UMI.CMS это объекты, которые являются документами в структуре сайта и таким образом связанны с иерархией в дереве сайта.

В следствии чего, у страниц есть такие атрибуты как:
- id в иерархии структуры
- id родительского элемента
- псевдостатический адрес
- title, h1, meta KEYWORDS, meta DESCRIPTIONS
- поля в группе полей «Дополнительные параметры»
- права на страницу
- и т. п.

При работе со страницами в кастомных макросах используйте классы относящиеся к модели иерархии и **id в иерархии структуры**.
Для работы с объектами используйте классы относящиеся к модели данных и **id объекта.**

Документация по классам модели иерархии доступна по адресу
http://api.umi-cms.ru/api.part.HierarchyModel.html

Документация по классам модели иерархии доступна по адресу
http://api.umi-cms.ru/api.part.DataModel.html

Правила использования системы прав доступа UMI.CMS

В системе можно управлять правами для:

- групп пользователей
- пользователей (*ни один пользователь не может иметь прав меньше, чем есть у пользователя "Гость")*
- страниц
- методов

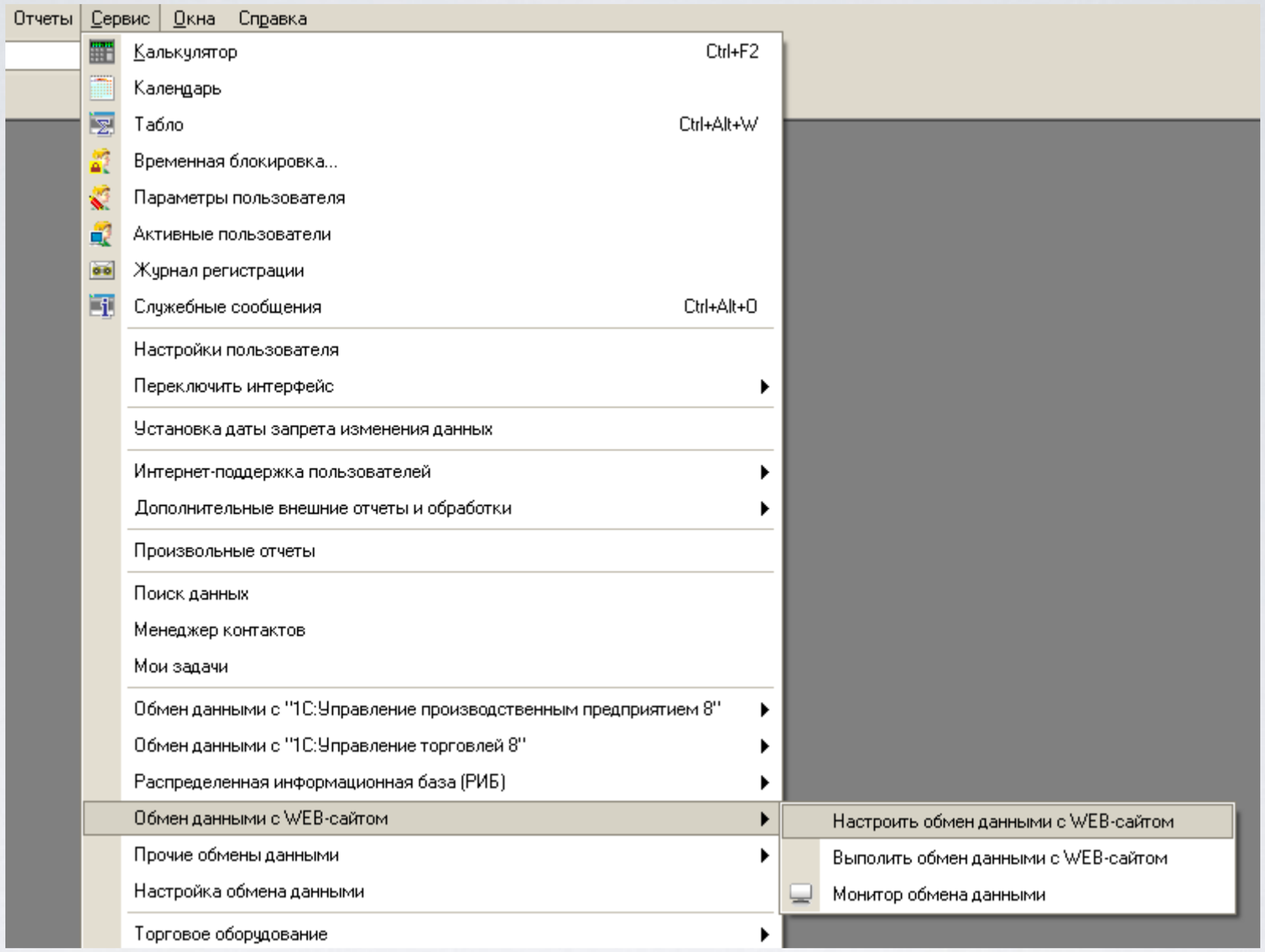
Если вы добавляете права в уже существующий модуль, то вносите изменения в файл **permissions.custom.php**, чтобы они не перезаписались при обновлении (*если такого файла нет в директории модуля, необходимо его создать).*

Пример файла **permissions.custom.php** для кастомного метода **%catalog custom_getCategoryList()%**

```php
<?php
    $permissions = Array(
        'view' => Array('custom_getCategoryList')
    );
?>
```

Интеграция UMI.CMS с программой 1С «Управление Торговлей»

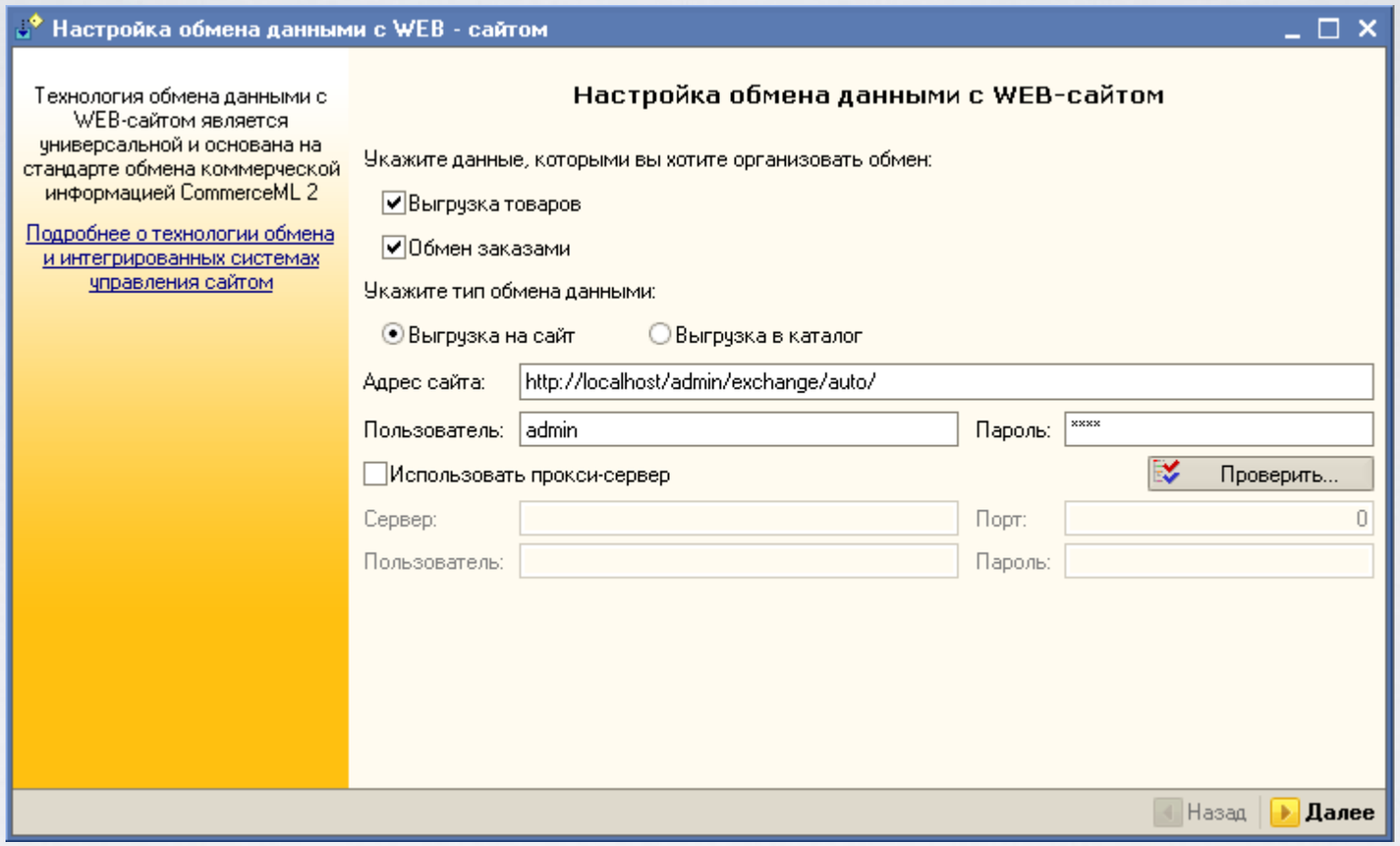
1. Настройка параметров синхронизации со стороны 1С (УТ)



Создадим новый обмен данных

Интеграция UMI.CMS с программой 1С «Управление Торговлей»

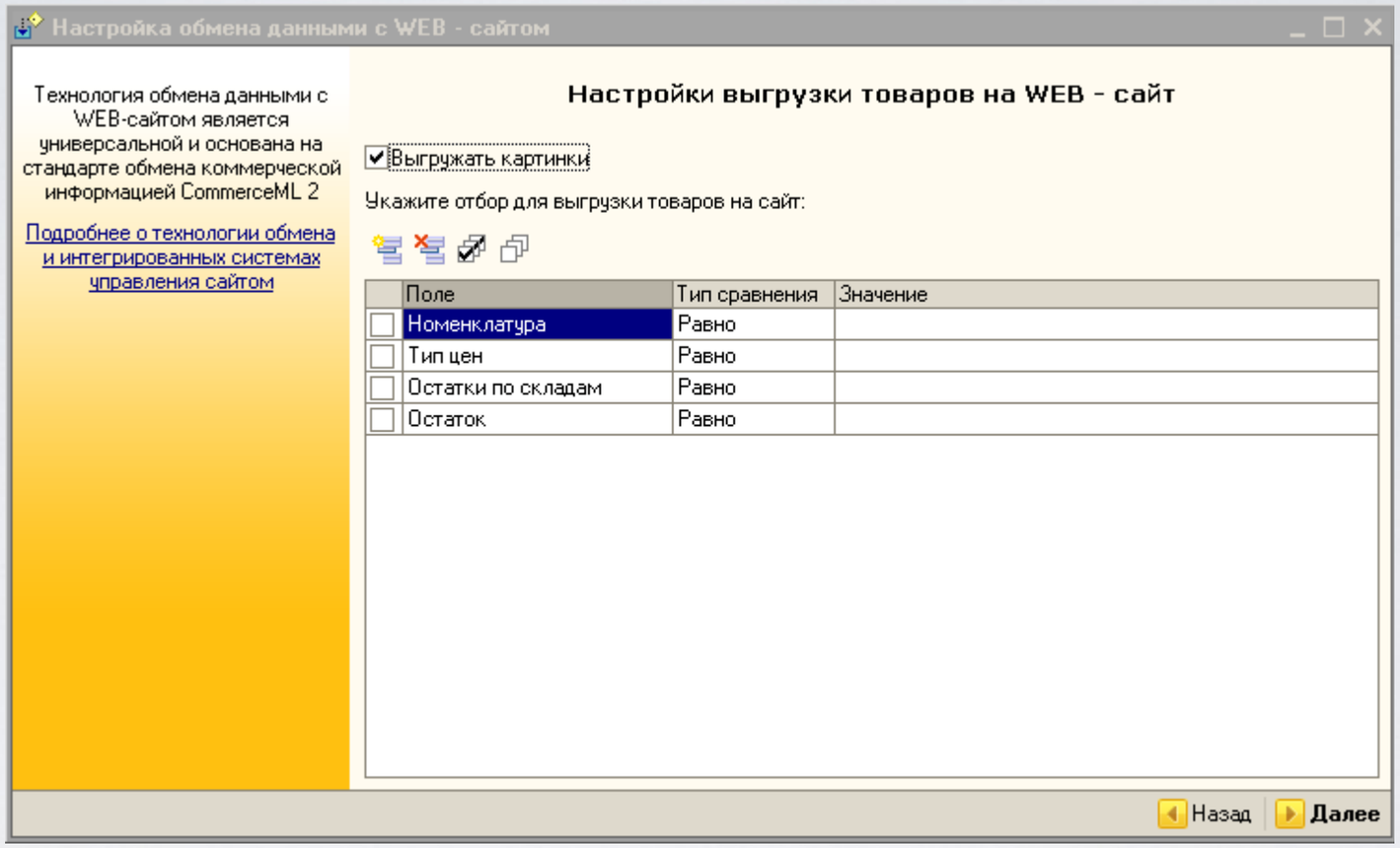
1. Настройка параметров синхронизации со стороны 1С (УТ)



Указываем ссылку на ваш сайт вида **ваш_сайт/admin/exchange/auto/** , логин, пароль и прочие параметры

Интеграция UMI.CMS с программой 1С «Управление Торговлей»

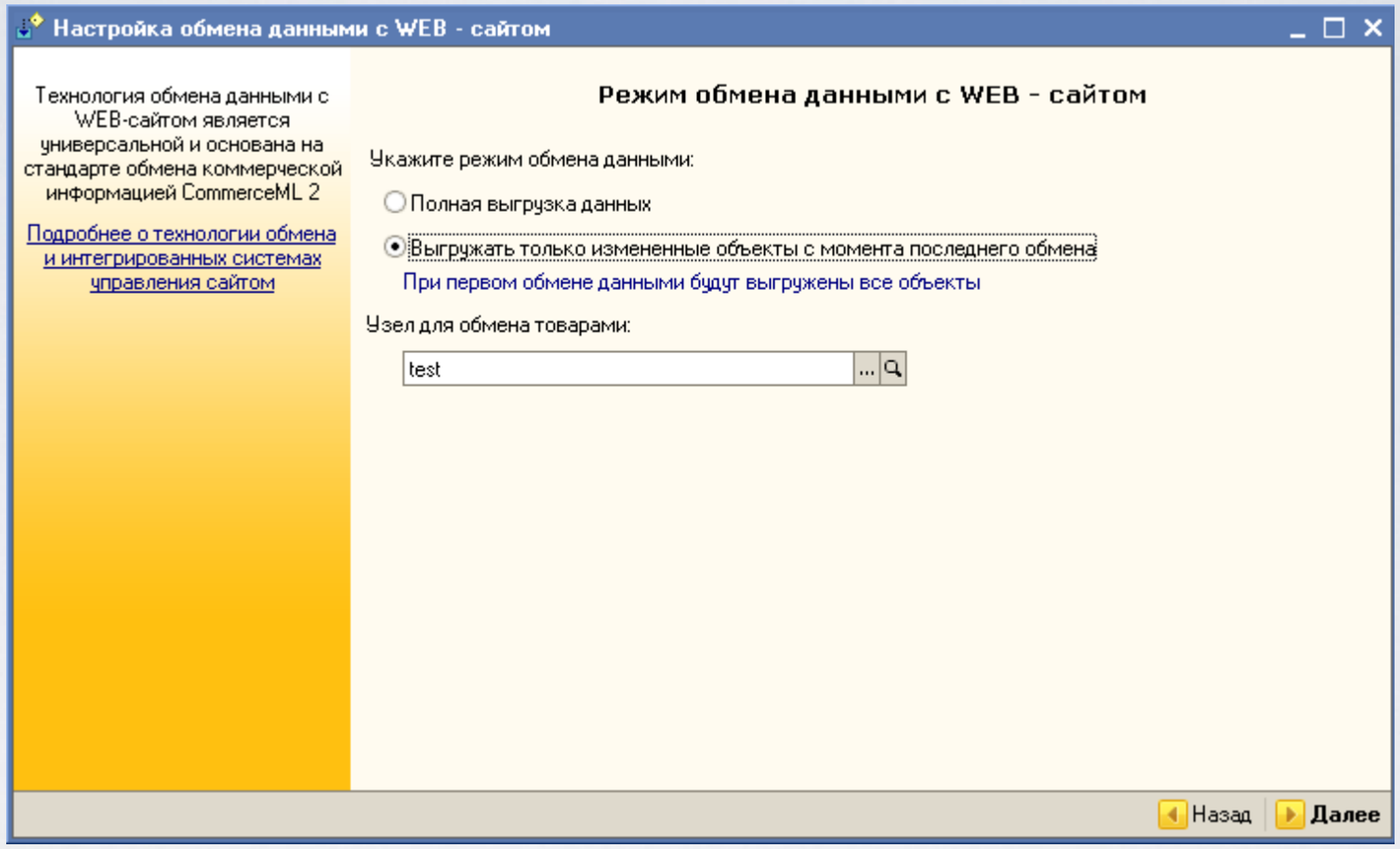
1. Настройка параметров синхронизации со стороны 1С (УТ)



Указываем какие товары мы хотим выгружать и выгружать ли картинки

Интеграция UMI.CMS с программой 1С «Управление Торговлей»

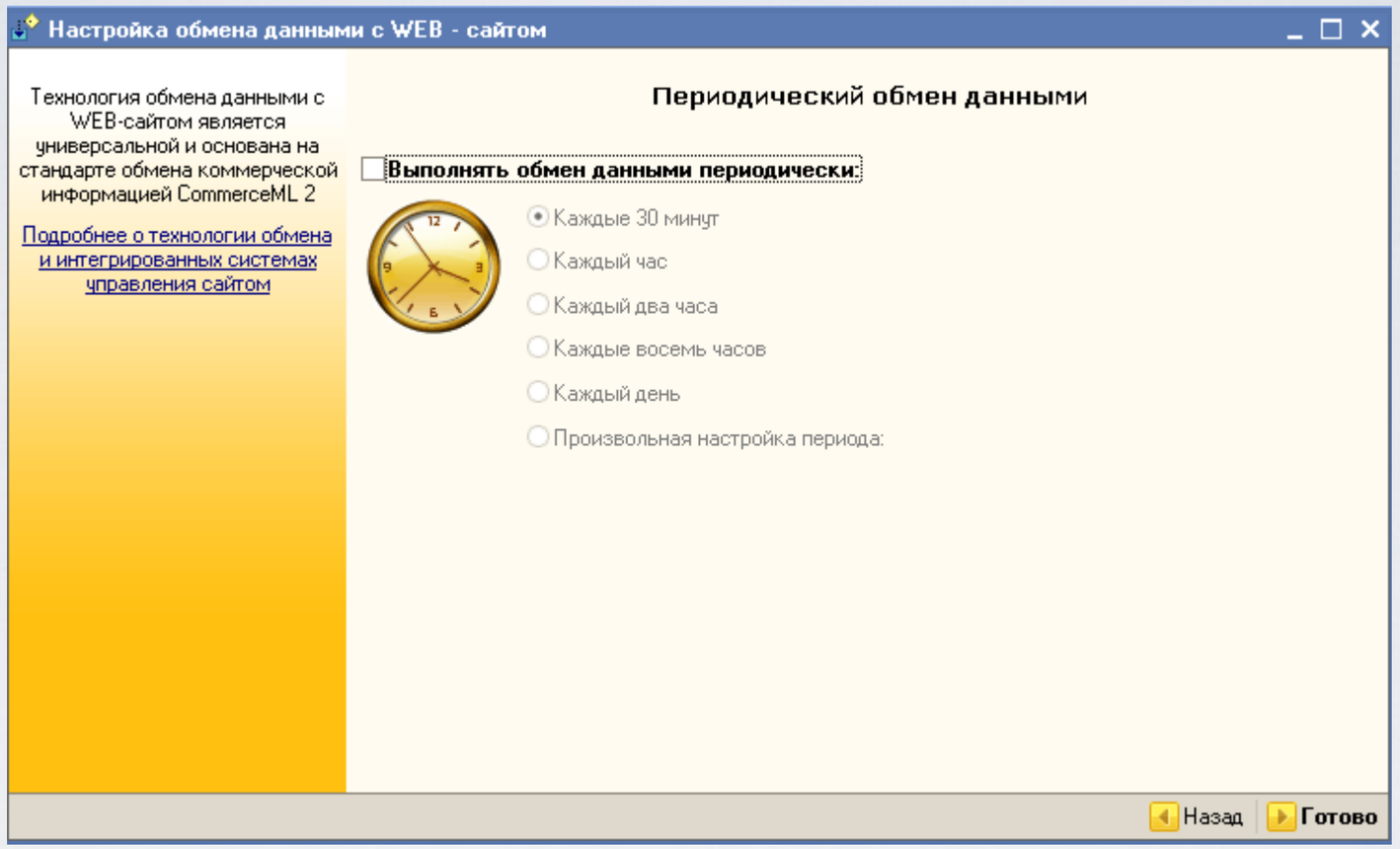
1. Настройка параметров синхронизации со стороны 1С (УТ)



Выгружать ли все товары или только измененные

Интеграция UMI.CMS с программой 1С «Управление Торговлей»

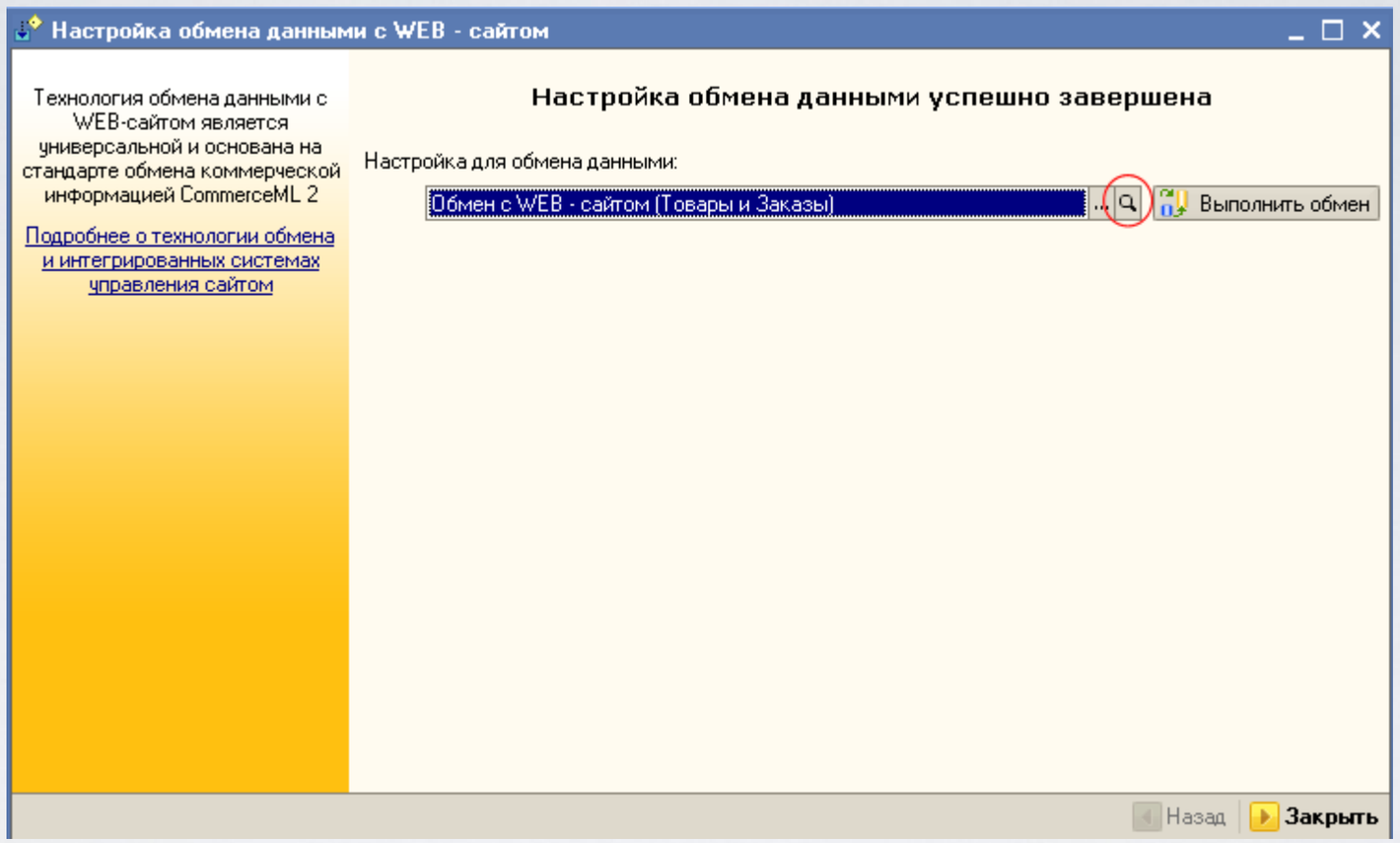
1. Настройка параметров синхронизации со стороны 1С (УТ)



Указываем периодичность выгрузки, если необходимо.

Интеграция UMI.CMS с программой 1С «Управление Торговлей»

1. Настройка параметров синхронизации со стороны 1С (УТ)



Настройка для обмена готова, если необходимо что-то изменить, нажимаем на «лупу».

Интеграция UMI.CMS с программой 1С «Управление Торговлей»

2. Общий принцип передачи данных и их обработки.

2.1. **Подготовка данных**. Система 1С подготавливает данные для отправки во временной директории операционной системы пользователя. В директории хранятся исходные данные, которые разбиты на равные части по 100Кб. Это части фалов import.xml, offers.xml и изображений (в том случае, если размер изображения более 100Кб).

2.2. **Передача данных**. Далее происходит отправка данных на сервер где установлена UMI.CMS. Метод auto() (находится файле classes/modules/exchange/__auto.php) принимает данные разбитые на части, собирает их . В папке sys-temp/1c_import/ собираются файлы import.xml и offers.xml. В папке /images/cms/data/ сохраняются изображения, обычно это папка import_files, в которой находятся вложенные папки с изображениями.

2.3. **Обработка данных**. Как только все данные в UMI.CMS получены, происходит обработка полученных данных. Для конвертации xml данных используется шаблон /xsl/import/commerceML2.xsl и /xsl/import/custom/commerceML2.xsl.

2.4. **Результаты импорта**. После импорта в структуре сайта появляются разделы каталога с объектами из 1С, а в модуле «Шаблоны данных», создаются дочерние типы данных по отношению к типу «Объект каталога»

Интеграция UMI.CMS с программой 1С «Управление Торговлей»

3. Настройка параметров синхронизации со стороны UMI.CMS.

В UMI.CMS есть возможность настроить импорт из 1С. Для этого в файле **/config.ini** существует секция [modules].

В этой секции можно:

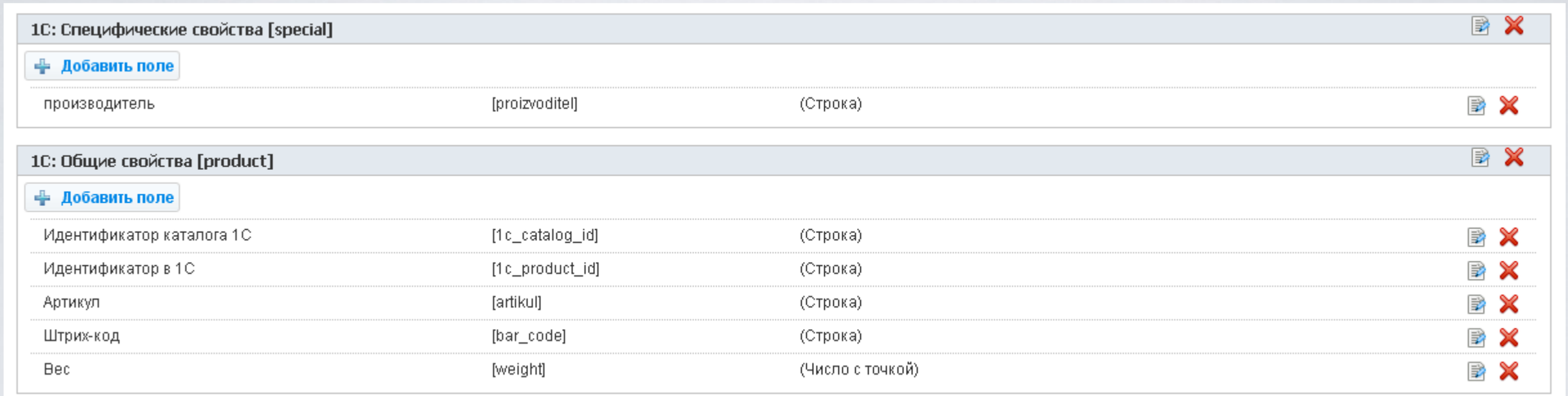
- назначать для объектов и разделов каталога конкретные шаблоны страниц (.tpl, .xsl);
- указывать будут ли включены объекты и разделы каталога после импорта;
- указывать будут ли выводиться в меню объекты и разделы каталога после импорта;
- указывать количество элементов выгружаемое за один раз при экспорте или импорте больших объемов;
- и т.д.

Документация по секции [modules]
http://help-dev.umi-cms.ru/part.Config.modules.html

Интеграция UMI.CMS с программой 1С «Управление Торговлей»

4. Вывод загруженных товаров на сайт.

Все свойства импортируемые из 1С попадают в группу полей «**1С: Специфические свойства**»


1С: Специфические свойства [special]

Добавить поле

производитель	[proizvoditel]	(Строка)

1С: Общие свойства [product]

Добавить поле

Идентификатор каталога 1С	[1c_catalog_id]	(Строка)
Идентификатор в 1С	[1c_product_id]	(Строка)
Артикул	[artikul]	(Строка)
Штрих-код	[bar_code]	(Строка)
Вес	[weight]	(Число с точкой)

При выводе свойств объекта каталога удобно пользоваться макросом **%data getPropertyGroup()%** , который выводит группу свойств страницы по определенному шаблону. Пример для нашего случая **%data getPropertyGroup(%id%,special)%**

Интеграция UMI.CMS с программой 1С «Управление Торговлей»

5. Выгрузка заказов в 1С (УТ).

После того, как в Интернет-магазине сделан заказ, для данного заказа автоматически ставится чекбокс "Выгружать заказ в 1С при следующем сеансе связи", отображение которого можно включить при табличном отображении заказов, на странице:

http://ваш_сайт/admin/emarket/orders/

В тот момент, когда запускается синхронизация с 1С, с активным параметром "Обмен заказами" в 1С, в нужный момент UMI.CMS формирует список заказов, которые необходимо выгрузить. Данные по заказам формируются в xml-формате CommerceML. После выгрузки заказов автоматически снимается чекбокс "Выгружать заказ в 1С при следующем сеансе связи". После того, как менеджером будет изменен "Статус заказа", чекбокс выгрузки снова будет установлен.

В самой 1С, в разделе "Документы – Продажи – Заказы покупателей" (путь для 1С версии 8.1), вы можете просмотреть все заказы, загруженные из UMI.CMS.

Рекомендации

При создании кастомных макросов помните:

- Файлы , которые вы меняете при разработке должны быть в кодировке utf-8.
- Кастомный макрос должен быть корректно написан и отлажен, чтобы не возникало критических ошибок, которые могут привести к неработоспособности сайта.
- Регистр имеет значение.
- Помните про права на выполнение макроса, если кастом располагается не в /classes/modules/custom.php, то по умолчанию результаты его работы доступны только супервайзеру.
- Для кастомных макросов и прочих изменений есть специальные файлы, все остальные файлы перезаписываются при обновлении системы.

СПАСИБО !

Даниил Сироткин

admin@umihelp.ru

www.umihelp.ru