


# Наценка на товары

Некоторые интернет-магазины предлагают различные опциональные дополнительные услуги для заказываемых товаров, эти услуги выбираются пользователем на этапе оформления заказа в корзине и влияют на окончательную стоимость заказа. Например, интернет-магазин по продаже какого-то сложного оборудования может предлагать услугу по настройке оборудования, за которую к товарам должна добавляться наценка.

## Shopping cart

Photo	Name		Qty	Price	Total	
	DCR-HC52E	<input checked="" type="checkbox"/> Test and set up	<input type="text" value="1"/>	7 703 \$	7 703 \$	<a href="#">✕</a>
Total:			1		7 703 \$	
<button>Update</button>			<button>Proceed checkout</button>			

Для реализации подобного механизма можно использовать данное решение.

Стандартного механизма для хранения наценки по каждому товару в корзине и заказе не предусмотрено, поэтому необходимо придумать механизм хранения этой информации. Изначально, когда заказ еще не создан будем хранить информацию о наценках в сессии, когда пользователь оформит заказ эта информация запишется в доп. свойство заказа и с помощью наблюдателей будет прикрепляться к товарам заказа. Цена товаров, для которых будет выбрана наценка, будет подменяться с помощью наблюдателя при выводе корзины, будут учитываться данные сессии или данные из доп. свойства заказа.

## Константы

Добавляем константу `MARKUP_FOR_TEST_PRODUCTS` со значением равным величину наценки.

## Доп. свойство для заказа

Идем в заказы интернет-магазина и добавляем новое доп. свойство с названием «Системная информация о наценках», типом «Большое текстовое поле» и XML-тегом `markups`, запрещаем множественные значения. Это поле будет использоваться решением для хранения информации о наценках на товары заказа.

## XSL-шаблон

В XSL-шаблон корзины добавить колонку для выбора товаров для наценки, например, так:

```

<div class="checkbox">
    <input type="checkbox" name="markup_{shop_item/@id}"
id="markup_{shop_item/@id}" value="1">
    <xsl:if test="/shop/markups/markup[shop_item_id =
current()/shop_item/@id]">
        <xsl:attribute name="checked" />
    </xsl:if>
</input>
<label for="markup_{shop_item/@id}">Test and set up</label>
</div>

```

## Наблюдатели

Для начала необходимо загрузить 4 наблюдателя.

### Наблюдатель для вывода наценки в заказе и ЦА (1)

- Загружаем файл: </modules/shop/order/item/observer/additionalfields.php>
- Добавляем в bootstrap.php:

```

Core_Event::attach('shop_order_item.onCallmarkup',
array('Shop_Order_Item_Observer_AdditionalFields', 'onCallmarkup'));

```

### Наблюдатель для подмены цены товара (2)

Загружаем файл: </modules/shop/item/controller/observer/addmarkuptoprice.php>

### Наблюдатель для прикрепления наценки к корзине (3)

Загружаем файл: </modules/shop/cart/controller/show/observer/attachmarkup.php>

## Формы центра администрирования

Идем в раздел Системы → Формы центра администрирования, находим форму «Список товаров в заказе», добавляем к ней поле новое поле «Наценка» с ключевым полем markup и типом «Вычисляемое поле (Используется обратный вызов функции)».

Это поле будет формироваться наблюдателем (1), подключенным в bootstrap.php и в центре администрирования должно выглядеть так:

Список товаров в заказе № 60

+ Добавить

20

Код	Наименование	Артикул	Кол-во	Цена	Наценка	Налог	Сумма
<input type="checkbox"/>							
<input type="checkbox"/>	119 NV-GS90		1.00	9083.00	—	0%	9083.00 \$
<input type="checkbox"/>	118 DCR-HC52E		1.00	7714.00	11 \$	0%	7714.00 \$

Удалить

\* Необходимо обратить внимание, что стоимость товара здесь выводится с учетом наценки, а наценка выступает в качестве информационного поля. Для вывода цены без наценки можно использовать метод `Shop_Order_Item_Observer_AdditionalFields::onCallprice_without_markup`, создав аналогично полю `markup` поле `price_without_markup`, а оригинальное поле с ценой из формы удалить. При этом в карточке товара в заказе все равно будет выводиться цена товара с наценкой.

## ТДС корзины

### Настройки страницы

1. Код вывода краткой корзины оборачиваем в наблюдатель:

```
Core_Event::attach('Shop_Item_Controller.onAfterCalculatePrice',
array('Shop_Item_Controller_Observer_AddMarkupToPrice',
'onAfterCalculatePrice'));
```

После вывода:

```
Core_Event::detach('Shop_Item_Controller.onAfterCalculatePrice',
array('
Shop_Item_Controller_Observer_AddMarkupToPrice',
'onAfterCalculatePrice'));
```

Должно быть так:

```
Core_Event::attach('Shop_Item_Controller.onAfterCalculatePrice',
array('Shop_Item_Controller_Observer_AddMarkupToPrice',
'onAfterCalculatePrice'));

$Shop_Cart_Controller_Show
->xsl(
```

```

        Core_Entity::factory('Xsl')->getByName(
            Core_Array::get(Core_Page::instance()->libParams,
                'littleCartXsl')
        )
    )
    ->couponText(Core_Array::get($_SESSION, 'coupon_text'))
    ->show();

Core_Event::detach('Shop_Item_Controller.onAfterCalculatePrice',
    array('Shop_Item_Controller_Observer_AddMarkupToPrice',
        'onAfterCalculatePrice'));

```

2. В настройках ТДС в коде пересчета/обновления корзины добавляем код для обработки добавленного чекбокса. Перед циклом добавляем:

```

$aMarkups = Core_Array::getSession('hostcmsCartMarkups', array());

```

Внутри цикла:

```

$markup = Core_Array::getRequest('markup_' .
    $oShop_Cart->shop_item_id);

if (!is_null($markup))
{
    $aMarkups[$oShop_Cart->shop_item_id] = MARKUP_FOR_TEST_PRODUCTS;
}
elseif (isset($aMarkups[$oShop_Cart->shop_item_id]))
{
    unset($aMarkups[$oShop_Cart->shop_item_id]);
}

```

После цикла:

```

$_SESSION['hostcmsCartMarkups'] = $aMarkups;

```

## Код страницы

1. Перед оператором switch для разделения на шаги:

```

Core_Event::attach('Shop_Item_Controller.onAfterCalculatePrice',
    array('Shop_Item_Controller_Observer_AddMarkupToPrice',
        'onAfterCalculatePrice'));

```

После:

```

Core_Event::detach('Shop_Item_Controller.onAfterCalculatePrice',
    array('Shop_Item_Controller_Observer_AddMarkupToPrice',
        'onAfterCalculatePrice'));

```

2. На 4-ом шаге внутри условия:

```
if ($_SESSION['hostcmsOrder']['shop_payment_system_id'])
{
```

Перед вызовом Shop\_Payment\_System\_Handler::execute добавляем:

```
$oShopOrderPropertyList =
Core_Entity::factory('Shop_Order_Property_List', $oShop->id);
$oProperty =
$oShopOrderPropertyList->Properties->getByTagName('markups');

if ($oProperty)
{
    $_SESSION['hostcmsOrder']['properties'][] = $aProperty =
array($oProperty->id, json_encode($_SESSION['hostcmsCartMarkups'],
JSON_PRETTY_PRINT));
}

Core_Event::attach('shop_order_item.onBeforeRedeclaredGetXml',
array('Shop_Order_Item_Observer_AdditionalFields',
'onBeforeRedeclaredGetXml'));
```

После вызова:

```
Core_Event::detach('shop_order_item.onBeforeRedeclaredGetXml',
array('Shop_Order_Item_Observer_AdditionalFields',
'onBeforeRedeclaredGetXml'));

$_SESSION['hostcmsCartMarkups'] = array();
```

3. Для ветки default перед выводом корзины добавляем:

```
Core_Event::attach('Shop_Cart_Controller_Show.onBeforeRedeclaredShow',
array('Shop_Cart_Controller_Show_Observer_AttachMarkup',
'onBeforeRedeclaredShow'));
```

После:

```
Core_Event::detach('Shop_Cart_Controller_Show.onBeforeRedeclaredShow',
array('Shop_Cart_Controller_Show_Observer_AttachMarkup',
'onBeforeRedeclaredShow'));
```

## Основной макет

Если в основном макете есть вывод краткой корзины, то его также нужно обернуть в следующий наблюдатель:

```
Core_Event::attach('Shop_Item_Controller.onAfterCalculatePrice',  
array('Shop_Item_Controller_Observer_AddMarkupToPrice',  
'onAfterCalculatePrice'));
```

После вывода:

```
Core_Event::detach('Shop_Item_Controller.onAfterCalculatePrice', array('  
Shop_Item_Controller_Observer_AddMarkupToPrice', 'onAfterCalculatePrice'));
```

Страницу создал Максим Засорин 23.01.17 в 12:33