Фильтрация по цене модификаций

В некоторых интернет-магазинах описание товаров предполагает активное использование модификаций, товары в этом случае задают базовое название и описание продукции, а модификации расширяют эти данные, дополняя различными характеристиками, например, объемом, весом, размером и т.д. Как правило, модификации в этом случае будут иметь различную стоимость, а базовые товары получат стоимость равную нулю.

При наличии большого количества таких товаров стоит задача правильной организации фильтрации в интернет-магазине. Стандратный фильтр позволяет организовать подход, при котором модификации выводятся на одном уровне с базовыми товарами, для этого необходимо задать:

```
$Shop_Controller_Show->modificationsList(TRUE);
```

В этом случае фильтрация товаров по цене будет работать корректно, но такой подход не всегда применителен. Иногда требуется подключить к фильтрации цены модификаций, но не выводить их на том же уровне, что и основные товары, реализация такого механизма есть в адаптивных фильтрах.

Если требуется организовать фильтрацию товаров по цене модификации без применения адаптивных фильтров, то можно использовать нижеприведенное решение, для его установки необходимо выполнить следующие шаги:

1. Для вывода правильных пределов в фильтре заменить:

   ```
   $Shop_Controller_Show->addMinMaxPrice();
   ```

   На:

   ```
   // Добавляем пределы для выборки цен с учетом модификаций
   $Shop_Controller_Show
       ->modificationsList(TRUE)
       ->addMinMaxPrice()
       ->modificationsList(FALSE);
   ```

2. В коде фильтра в ТДС Интернет-магазин перед выборкой `absolute_price` добавить следующий код, он повторяет выборку товаров, но теперь выбирается цена модификации, попадающая в заданный фильтром интервал:

   ```
   // Выбираем цену модификации, попадающую в нужный предел
   $oModificationsWithPrice = Core_QueryBuilder::select()
   ->select(array(Core_QueryBuilder::expression(str_replace('shop_items',
   'modifications', $query_currency_switch)), 'absolute_price'))
       ->from(array('shop_items', 'modifications'))
       ->leftJoin('shop_item_discounts', 'modifications.id', '=',
   'shop_item_discounts.shop_item_id')
       ->leftJoin('shop_discounts',
   'shop_item_discounts.shop_discount_id', '=', 'shop_discounts.id',
   array(
   ```

```
        array('AND ' => array('shop_discounts.active', '=', 1)),
        array('AND ' => array('shop_discounts.deleted', '=', )),
        array('AND' => array('shop_discounts.start_datetime', '<=',
$current_date)),
        array('AND (' => array('shop_discounts.end_datetime', '>=',
$current_date)),
        array('OR' => array('shop_discounts.end_datetime', '=',
'0000-00-00 00:00:00')),
        array(')' => NULL)
    ))
    ->where('modifications.modification_id', '=',
Core_QueryBuilder::expression('`shop_items`.`id`'))
    ->groupBy('modifications.id')
    ->limit(1)
;

if ($price_from)
{
    $oModificationsWithPrice->having('absolute_price', '>=',
$price_from);
}

if ($price_to)
{
    $oModificationsWithPrice->having('absolute_price', '<=',
$price_to);
}

// Добавляем кастомный столбец
$Shop_Controller_Show->shopItems()
    ->queryBuilder()
        ->select(array($oModificationsWithPrice,
'modification_price'));

// Функция обратного вызова для кастомного столбца
class Shop_Item_Observer_ModificationPrice
{
    static public function onCallmodification_price()
    {

    }
}

Core_Event::attach('shop_item.onCallmodification_price',
array('Shop_Item_Observer_ModificationPrice',
'onCallmodification_price'));
```

3. Затем после установки условий на выборку цены добавить такой код:

```
if ($price_from || $price_to)
{
```

```
    $Shop_Controller_Show->shopItems()->queryBuilder()
        ->having('absolute_price', '>', )
        ->setOr()
        ->having('modification_price', '>', );
}
```

4. Ниже для корректной сортировки после:

```
->orderBy('absolute_price', $sorting == 1 ? 'ASC' : 'DESC')
```

Добавить:

```
->orderBy('modification_price', $sorting == 1 ? 'ASC' : 'DESC')
```

Страницу создал Максим Засорин 23.12.16 в 11:05