
django-lfs Documentation

Release dev

Kai Diefenbach

Mar 09, 2017

Contents

1	Overview	1
2	Installation	3
3	Upgrade notes	9
4	Getting Started	11
5	Information for Users	15
6	Information for Developers	89
7	Miscellaneous Information	125
8	Indices and Tables	127

What is LFS?

LFS is an online shop based on widely-used software: Python, Django and jQuery. It is open source, free, easy to use, secure and fast.

Features

For an up-to-date list of features please visit the official website: <http://www.getlfs.com/features>.

How To Read This Documentation

You should start with the *concepts of LFS* to get an overview what it can do and what not. After that you might go through the *getting started tutorial*. Once you have done this you could try to enter the data you like. When you stuck look into the *how- to's for users* or refer to the *management interface reference*.

Note: If you haven't installed a shop yet you may use the [demo shop](#) to try things out. Please be aware that we reset it every two hours.

Information

There are several places where you can find additional information and help:

- [Download on PyPi](#)
- [Demo shop](#)

- [Source code](#)
- [Google Group](#)
- [Twitter](#)
- [IRC channel](#)
- [Professional support](#)

Prerequisites

Make sure you have installed:

- Python 2.6.x or 2.7.x
- A RDBMS of your choice (PostgreSQL, MySQL, SQLite or Oracle)
- libjpeg8-dev zlib1g-dev (for Image handling via Pillow)

Note: Be prepared that we might abandon all other databases but PostgreSQL in one of the next releases.

Installation

The installation is straightforward and should last just a few minutes. Please execute following steps:

1. Preferably create a virtualenv and activate it, e.g.:

```
$ virtualenv lfs_env
```

2. Install django-lfs:

```
$ pip install django-lfs
```

3. If you not have an existing one, create a new project:

```
$ django-admin startproject <your-projectname>
```

4. Add the following urls to your project's `urls.py` file:

```
import django.views.static
from django.conf import settings
from django.conf.urls import include

url(r'', include('lfs.core.urls')),
url(r'^manage/', include('lfs.manage.urls')),
url(r'^reviews/', include('reviews.urls')),
url(r'^media/(?P<path>.*)$', django.views.static.serve, {'document_root':
↳ settings.MEDIA_ROOT})),
```

5. Add the following settings to your project's `settings.py` file:

```
# Django
from django.utils.translation import ugettext_lazy as _

INSTALLED_APPS = [
    'lfs_theme',
    ...
    'django.contrib.redirects',
    'django.contrib.sites',
    'compressor',
    'lfs.addresses',
    'lfs.caching',
    'lfs.cart',
    'lfs.catalog',
    'lfs.checkout',
    'lfs.core',
    'lfs.criteria',
    'lfs.customer',
    'lfs.customer_tax',
    'lfs.discounts',
    'lfs.export',
    'lfs.gross_price',
    'lfs.mail',
    'lfs.manage',
    'lfs.marketing',
    'lfs.manufacturer',
    'lfs.net_price',
    'lfs.order',
    'lfs.page',
    'lfs.payment',
    'lfs.portlet',
    'lfs.search',
    'lfs.shipping',
    'lfs.supplier',
    'lfs.tax',
    'lfs.tests',
    'lfs.utils',
    'lfs.voucher',
    'lfs_contact',
    'lfs_order_numbers',
    'lfs_paypal',
    'localflavor',
    'paypal.standard.ipn',
    'portlets',
    'postal',
    'reviews',
]
```

```

AUTHENTICATION_BACKENDS = (
    'lfs.customer.auth.EmailBackend',
    'django.contrib.auth.backends.ModelBackend',
)

MEDIA_URL = '/media/'
MEDIA_ROOT = BASE_DIR + '/media'

SESSION_SERIALIZER = "django.contrib.sessions.serializers.PickleSerializer"

SITE_ID = 1

STATICFILES_FINDERS = (
    'django.contrib.staticfiles.finders.AppDirectoriesFinder',
    'compressor.finders.CompressorFinder',
)

STATIC_URL = '/static/'
STATIC_ROOT = BASE_DIR + "/sitestatic"

# Please note that ``lfs.core.context_processors.main`` has been added
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
                'lfs.core.context_processors.main',
            ],
        },
    },
]

# LFS
# see http://docs.getlfs.com/en/latest/developer/settings.html for more
LFS_AFTER_ADD_TO_CART = "lfs_added_to_cart"
LFS_RECENT_PRODUCTS_LIMIT = 5

LFS_CRITERIA = [
    ["lfs.criteria.models.CartPriceCriterion", _(u"Cart Price")],
    ["lfs.criteria.models.CombinedLengthAndGirthCriterion", _(u"Combined Length_
↪and Girth")],
    ["lfs.criteria.models.CountryCriterion", _(u"Country")],
    ["lfs.criteria.models.HeightCriterion", _(u"Height")],
    ["lfs.criteria.models.LengthCriterion", _(u"Length")],
    ["lfs.criteria.models.WidthCriterion", _(u"Width")],
    ["lfs.criteria.models.WeightCriterion", _(u"Weight")],
    ["lfs.criteria.models.ShippingMethodCriterion", _(u"Shipping Method")],
    ["lfs.criteria.models.PaymentMethodCriterion", _(u"Payment Method")],
]

LFS_ORDER_NUMBER_GENERATOR = "lfs_order_numbers.models.OrderNumberGenerator"

```

```

LFS_DOCS = "http://docs.getlfs.com/en/latest/"

LFS_INVOICE_COMPANY_NAME_REQUIRED = False
LFS_INVOICE_EMAIL_REQUIRED = True
LFS_INVOICE_PHONE_REQUIRED = True

LFS_SHIPPING_COMPANY_NAME_REQUIRED = False
LFS_SHIPPING_EMAIL_REQUIRED = False
LFS_SHIPPING_PHONE_REQUIRED = False

LFS_PAYMENT_METHOD_PROCESSORS = [
    ["lfs_paypal.processor.PayPalProcessor", _(u"PayPal")],
]

LFS_PRICE_CALCULATORS = [
    ['lfs.gross_price.calculator.GrossPriceCalculator', _(u'Price includes tax')],
    ['lfs.net_price.calculator.NetPriceCalculator', _(u'Price excludes tax')],
]

LFS_SHIPPING_METHOD_PRICE_CALCULATORS = [
    ["lfs.shipping.calculator.GrossShippingMethodPriceCalculator", _(u'Price_
↪includes tax')],
    ["lfs.shipping.calculator.NetShippingMethodPriceCalculator", _(u'Price_
↪excludes tax')],
]

LFS_UNITS = [
    _(u"l"),
    _(u"m"),
    _(u"cm"),
    _(u"lfm"),
    _(u"Package(s)"),
    _(u"Piece(s)"),
]

LFS_PRICE_UNITS = LFS_BASE_PRICE_UNITS = LFS_PACKING_UNITS = LFS_UNITS

# Paypal
LFS_PAYPAL_REDIRECT = True
PAYPAL_RECEIVER_EMAIL = "info@yourbusiness.com"
PAYPAL_IDENTITY_TOKEN = "set_this_to_your_paypal_pdt_identity_token"

# Reviews
# see http://django-reviews.readthedocs.io/en/latest/#settings for more
REVIEWS_SHOW_PREVIEW = False
REVIEWS_IS_NAME_REQUIRED = False
REVIEWS_IS_EMAIL_REQUIRED = False
REVIEWS_IS_MODERATED = False

```

Optionally you might add:

```

# Django
# see https://docs.djangoproject.com/en/1.10/topics/cache/ for more.
CACHES = {
    'default': {
        'BACKEND': 'django.core.cache.backends.locmem.LocMemCache',
    },
}

```

```
# Compressor
# see http://django-compressor.readthedocs.io/en/latest/settings/ for more
COMPRESS_CSS_FILTERS = [
    'compressor.filters.css_default.CssAbsoluteFilter',
    'compressor.filters.cssmin.CSSCompressorFilter',
]
COMPRESS_ENABLED = True
COMPRESS_OFFLINE = True
```

1. \$ python manage.py migrate
2. \$ python manage.py lfs_init
3. \$ python manage.py runserver
4. Browse to <http://localhost:8000>

Note: If you encounter problems, please see trouble shooting.

Note: If you're setting up a production environment then you should not use Django's builtin development server (bin/django runserver). Instead, you'll probably want to use uWsgi or Gunicorn servers. Check Django (and uWsgi/Gunicorn) documentation for details.

Note: For production environments you're supposed to change robots.txt file (otherwise bots/crawlers like google bot will not be allowed to scan your site, which is not what you probably want). Default version of robots.txt is located at lfs_theme application: templates/lfs/shop/robots.txt. You should create your own 'mytheme' app with structure like: templates/lfs/shop/robots.txt and place it in settings.INSTALLED_APPS before(!) 'lfs_theme'. Also note, that in production environment it is good to serve robots.txt directly from HTTP server like nginx or Apache.

Migration from version 0.9 to version 0.10 and higher

Migration starting from 0.10 is based on Django's default migrations, see: <https://docs.djangoproject.com/en/1.8/topics/migrations/>

1. Install the new LFS version
2. Backup your existing database
3. Enter your existing database to lfs_project/settings.py
4. \$ bin/django migrate

Migration from versions 0.5 - 0.8 to version 0.10

Migrations from 0.5 - 0.8 to version 0.10 needs an intermediate step through version 0.9.

Migration from versions 0.5 - 0.8 to version 0.9

Migration from versions 0.5 - 0.8 to version 0.9 can be done with a migration command (`lfs_migrate`) which migrates existing databases up to version 0.9.

1. Install the 0.9
2. Backup your existing database
3. Enter your existing database to `lfs_project/settings.py`
4. `$ bin/django syncdb`
5. `$ bin/django lfs_migrate`

What's next?

Move on to *Getting Started*.

Below is a description of changes that are useful to know about when upgrading LFS to newer version

Upgrading from 0.7.x to 0.8.x

This is not full reference of changes but at least some of them are described:

- currency templatetag always returns HTML now, eg. `Fr. 999`, previously HTML was only returned for negative values
- currency templatetag in text email templates (`lfs_theme/templates/lfs/mail/order_details.txt`) was removed in favour of `currency_text` templatetag. The latter one doesn't return HTML (ever).
- `lfs/base.html` has slightly different structure to the footer and colophon sections due to incorrect width of these elements in previous layout. `div.colophon-inner` and `div.footer-inner` html elements were added, both with `padding: 10px` set in `main.css`. `padding: 10px` was removed from `'#footer .container'` and `'#colophon .container'` in `main.css`
- `update_editor` method in `lfs_tinymce.js` has been modified and requires tinymce 3.5.8 which is now being used. References to tinymce were changed in `manage_base.html` and `lfs_tinymce.js`
- filter portlet has been updated to allow for manufacturer filtering and because of that its template: `lfs_theme/templates/lfs/portlets/filter.html` was modified - manufacturer filtering section has been added
- Small change at `lfs/templates/manage/product/product.html` - removed `onkeypress` from filter input element in favour of css class `'disable-enter-key'`. Changed `lfs.manage.js` to add event handler for `'disable-enter-key'`.
- Added new ORDER state: PREPARED that can be used to mark orders as prepared to be sent.
- Added new signal and setting that allows defining extra ORDER_STATES. Signal is `order_state_changed` and option is `LFS_EXTRA_ORDER_STATES`. New states should start with id 20 or higher to avoid conflicts with built in states.
- Use `'SHOP'` instead of `'shop'` in `lfs/shop/shop.html`

- Added position column to PropertyGroups and ability to order these with drag & drop in management panel - modified lfs.manage.js and management template for property groups.
- Added LFS_CHECKOUT_NOT_REQUIRED_ADDRESS setting. This allows to change address that is not required at checkout page. Changed one_page_checkout.html template, lfs.js and OnePageCheckout Form.
- refactored lfs.manage.js - do not use live anymore. Updated manage/export/export.html, manage/export/export_inline.html, manage/manufactuers/manufacturer.html and manage/manufacturers/manufacture_inline.html to use data-url instead of just 'data' and use elem.data('something') in JavaScript
- added <div id="portlets-dialog" title="{% trans "Portlets dialog" %}"></div> to manage_base.html to handle properly inserting images to TinyMCE within portlets dialog (changes to lfs.manage.js with portlets dialog)
- added some SEO related attributes to templates and canonical tags for variants
- modified catalog/views.py -> category_products and catalog/views.py -> category_categories return value, so that it now contains pagination data for use in main template (SEO optimization with rel="next/prev" (template: lfs/catalog/category_base.html)
- modified mimetype returned by ajax calls to: application/json. This requires changes in javascript ajax calls: lfs.js, lfs.manage.js, lfs_tinymce.js, manage/product/attachments.html(!)
- moved javascript code from manage/product/images.html to lfs.manage.product.js and updated to use proper mimetypes in responses

CHAPTER 4

Getting Started

This document explains the first steps after the shop has been installed. For the installation process please refer to *Installation*. If you want to know more about single data fields within the forms below, you can just click on the **Help** menu, which opens the context aware help.

Note: If you don't have a working LFS instance yet, you can just use our demo shop at <http://demo.getlfs.com>. Please be aware that we reset the database every two hours.

Enter Shop Preferences

1. Browse <http://localhost:8000/manage>.
2. Login, if you aren't yet.
3. Browse to Shop / Preferences.
4. Go to the Shop tab.
5. Enter the Name. This is the name of the shop, for instance `ACME Inc` which is used on several places, e.g. as part of the meta title of all HTML pages.
6. Enter the Shop Owner. This is the full name of the shop owner.
7. Enter From E-mail Address. This e-mail address is used as sender for all e-mails which are sent from the shop, e.g. the order confirmation mail to the shop customer.
8. Enter Notification E-Mail Addresses. To this addresses all notification messages will be sent. For example, if an order has been submitted.
9. Click on the Save Shop button.
10. Go to the Default Values tab.
11. Select the default Price Calculator:. This is the default for all products, if the price calculator is not explicitly selected on a product.

12. Select `Invoice Countries` and `Shipping Countries`. All selected countries will be available to the shop customers for selection.
13. Select `Default Country`. This country will be preselected as default country.
14. Click on the `Save Default Values` button.

Add Product Taxes

1. Go to `Shop / Product Taxes` and click on the `Add Product Tax` button.
2. Enter the `Rate` (in percentage), for instance 19.0.
3. Click on the `Add Tax` button.
4. You can add more taxes by clicking on the `Add Product Tax` site action and executing steps 2 to 3.

Add Delivery Times

1. Go to `Shop / Delivery Times` and click on the `Add Delivery Time` button.
2. Enter `Min`, the minimal delivery time.
3. Enter `Max`, the maximal delivery time.
4. Enter the `Unit` of this delivery time.
5. Click on the `Add Delivery Time` button.
6. You can add more delivery times by clicking on `Add delivery time` link and executing steps 2 to 3.

Add Categories

1. Now go to `Catalog / Categories`.
2. Click on the `Add Category` button.
3. Enter a name and a slug.
4. Click on the `Add Category` button.
5. Go to the `Data` tab.
6. Enter a `Description`. This will be displayed within the detail view of a category.
7. Click on the `Save Data` button.
8. Go to the `View` tab.
9. Select the `Category Template`, In this case `Category with Products` which means the assigned products of the category will be displayed.
10. Click on the `Save View` button.

Add Products

1. Now go to `Catalog / Products` and click on the `Add Product` button.
2. Enter `name` and a `slug`.
3. Click on the `Add Product` button.
4. Go to the `Data` tab
5. Enter the `SKU` of the product. This is the unique id of the product - taken from your ERP for instance.
6. Enter the `Price` of the product.
7. Enter a `Short Description`. This will be displayed when the product is displayed within a overview, e.g. when a category displays it's assigned products.
8. Enter a `Description`. This will be displayed within the detail view of a product.
9. Click on the `Save Data` button.
10. Go to the `Categories` tab
11. Select the above entered category and click `Save Categories`.
12. Go to the `Images` tab
13. Click `Choose Files`, browse to your images and select all images you want to upload. You will see an upload indicator and all images will be uploaded.
14. Go back to the `Data` tab
15. Check the `Active` check box. Only active products are displayed to the customers.
16. Click on the `Save Data` button.
17. Click on `Goto Product` to visit the new product.

Set Default Locale and Currency

Go to `lfs_project/settings.py` and change `LFS_LOCALE` to your needs (the default one is `en_US.utf8`). This will activate the correct currency and number formats at the same time.

Usually there are several locale installed on your computer. In order to check which ones, please open a terminal and type:

```
locale -a
```

To install an english locale (on Debian/Ubuntu) please enter:

```
sudo apt-get install language-support-en
```

Note: After you have changed the locale you need to restart your instance to make it active.

See also:

- <http://en.wikipedia.org/wiki/Locale>
- <http://docs.python.org/library/locale.html>

What's Next?

- Add more categories and products.
- *Add accessories to your products.*
- *Add related products to your products.*
- *Add variants.*
- *Manage taxes.*
- *Manage shipping methods.*
- *Manage payment methods.*
- *Manage delivery times.*
- *Manage stock information.*
- Add some portlets to your shop and/or categories.

Concepts

Products

This section describes the different product types of LFS.

Overview

Products are the most essential content object of LFS. They are goods which are sold to customers. LFS' products manage a lot of data which are relevant for an online shop. Please see the [product management interface reference](#) to see all available data fields in detail. LFS provides three types of products: default products, configurable products and products with variants. They are described in more detail in the following sections.

Default Product

This is the simple and straightforward product of LFS. All other product types are based on the default product. It handles a lot of information like:

- General descriptions
- Prices and price units
- Images
- Attachments
- *Accessories*
- *Related products*
- Stock data
- SEO Data

- [*Portlets*](#)
- [*Properties*](#)

Each of this is described in more detail throughout this documentation.

Configurable Product

A configurable product is a product with several properties and options from which a customer can or needs to choose. For instance a property `Color` and its options `red`, `yellow` and `green`. The selected options of the properties can change the total price of the product, which is calculated by the base price plus the prices of the selected options.

See also:

- [*How To Add a Configurable Product*](#)

Product with Variants

A `product with variants` is a product from which the customer can choose out of an arbitrary amount of similar variants. It consists out of two parts: the base and the variants.

Base

The base can't be sold to customers, but it serves as a container for the variants and provides default data, which can be inherited from the variants.

To create variants for a base *global* and *local properties* are used, e.g. the property `Color` and its options `red`, `yellow` and `green`. Each variant of a base belongs to a unique combination of options of all properties of a base.

Variant

The variants can be sold to a customer. Each variant is a discrete product with its own data, e.g. own price, name and SKU. By default the variants inherit all data from the base. This data can be overwritten per variant and field.

See also:

- [*How To Add a Product with Variants*](#)

See Also

- [*Product Management Interface*](#)

Categories

This section describes the concepts of categories.

Overview

Categories are used to structure the *products* of the shop.

Every category can have either one or no parent category. In this way the category tree is built, which serves as the essential navigation of the shop. If a category has no parent category, it is considered a top level category.

Every category can have an arbitrary amount of products and/or sub categories. What of these are displayed depends on the selected template. How these are displayed depends on several format information of a category.

Each category can have several assigned *portlets*. By default the portlets are inherited from the parent category or from the shop preferences (in case of top level categories). This can be blocked per slot.

In addition a *static block* can be assigned to a category, which is displayed on top of the category page.

Please see the description of the *Category Management Interface* in order to see more details of the information categories provide.

See Also

- *Category Management Interface*

Manufacturers

This section describes the concepts of manufacturers.

Overview

Manufacturers are bound directly to the *products* that are sold at the shop.

Every manufacturer can have zero or more products assigned. Each product can have zero or one manufacturer.

There is a page that lists all the manufacturers defined in the system. In order to use it at your shop you have to add new *action* pointing to **/manufacturers** url. From this page, users can dig deeper to see all the products from the specific manufacturer.

Please see the description of the *Manufacturers Management Interface* in order to see more details of the information manufacturers provide.

See Also

- *Manufacturers Management Interface*

Criteria

This section describes the concepts of criteria.

Overview

Criteria are a central concept of LFS and are used on several places in order to restrict or display information on base of the current situation. For example: which products are in the cart or in which country are these products about to be delivered. Criteria have a value and an operator, which are the base whether the criteria is true or false. This is described in more detail in the section *Criteria* below.

Criteria are used within:

- Payment methods
- Payment method's prices
- Shipping methods
- Shipping method's prices
- Discounts

For example a shipping method is only available when all of its criteria are true.

Criteria

This paragraph describes the existing types of criteria and which operators they provide.

Cart Price

Description

Tests the gross price of all products within the cart (without costs for shipping, payment and so on).

Value

A number against the gross price of all products within the cart is tested.

Operators

Equal The criterion is true, if the gross price of the cart is equal to the entered value.

Less than The criterion is true, if the gross cart price is less than the entered value.

Less than equal to The criterion is true, if the gross cart price is less than or equal to the entered value.

Greater than The criterion is true, if the gross cart price is greater than the entered value.

Greater than equal to The criterion is true, if the gross cart price is greater than or equal to the entered value.

Payment Method

Description

Provides some tests for the payment methods of the shop.

Value

Any selection out of all provided payment methods.

Operators

Is selected The criterion is true if the current selected payment method is within the selected payment methods.

Is not selected The criterion is true if the current selected payment method is not within the selected payment methods.

Is valid The criterion is true if all of the selected payment methods are valid.

Is not valid The criterion is true if all of the selected payment methods are not valid.

Shipping Method

Description

Provides some tests for the shipping methods of the shop.

Value

Any selection out of all provided shipping methods.

Operators

Is selected The criterion is true if the current selected shipping method is within the selected shipping methods.

Is not selected The criterion is true if the current selected shipping method is not within the selected shipping methods.

Is valid The criterion is true if all of the selected shipping methods are valid.

Is not valid The criterion is true if all of the selected shipping methods are not valid.

Country

Description

Tests the country of the customer's shipping address.

Value

An arbitrary selection out of all existing countries.

Operators

Is The criterion is true, if the customer's shipping country is within the selection of countries.

Is not The criterion is true, if the customer's shipping country is not within the selection of countries.

Combined Length And Girth

Description

Tests the total combined length and girth (clag) of all products within the cart. The clag is calculated as following:

```
(2 * maximum width) + (2 * total height) + maximal length
```

Value

A number against the total combined length and girth of all products within the cart is tested.

Operators

Equal The criterion is true, if the total clag is equal to the entered value.

Less than The criterion is true, if the total clag is less than the entered value.

Less than equal to The criterion is true, if the total clag is less than equal to the entered value.

Greater than The criterion is true, if the total clag is greater than the entered value.

Greater than equal to The criterion is true, if the total clag is greater than equal to the entered value.

Height

Description

Tests the total height of all products within the cart.

Value

A number against the total height of all products within the cart is tested.

Operators

Equal The criterion is true, if the total height of all products within the cart is equal to the entered value.

Less than The criterion is true, if the total height of all products within the cart is less than the entered value.

Less than equal to The criterion is true, if the total height of all products within the cart is less than equal to the entered value.

Greater than The criterion is true, if the total height of all products within the cart is greater than the entered value.

Greater than equal to The criterion is true, if the total height of all products within the cart is greater than equal to the entered value.

Length

Description

Tests the maximum length of all products within the cart.

Value

A number against the maximum length of all products within the cart is tested.

Operators

Equal The criterion is true, if the maximum length of all products within the cart is equal to the entered value.

Less than The criterion is true, if the maximum length of all products within the cart is less than the entered value.

Less than equal to The criterion is true, if the maximum length of all products within the cart is less than equal to the entered value.

Greater than The criterion is true, if the maximum length of all products within the cart is greater than the entered value.

Greater than equal to The criterion is true, if the maximum length of all products within the cart is greater than equal to the entered value.

Weight

Description

Tests the total weight of all products within the cart.

Value

A number against the total weight of all products within the cart is tested.

Operators

Equal The criterion is true, if the total weight of all products within the cart is equal to the entered value.

Less than The criterion is true, if the total weight of all products within the cart is less than the entered value.

Less than equal to The criterion is true, if the total weight of all products within the cart is less than equal to the entered value.

Greater than The criterion is true, if the total weight of all products within the cart is greater than the entered value.

Greater than equal to The criterion is true, if the total weight of all products within the cart is greater than equal to the entered value.

Width

Description

Tests the maximum width of all products within the cart.

Value

A number against the maximum width of all products within the cart is tested.

Operators

Equal The criterion is true, if the maximum width of all products within the cart is equal to the entered value.

Less than The criterion is true, if the maximum width of all products within the cart is less than the entered value.

Less than equal to The criterion is true, if the maximum width of all products within the cart is less than equal to the entered value.

Greater than The criterion is true, if the maximum width of all products within the cart is greater than the entered value.

Greater than equal to The criterion is true, if the maximum width of all products within the cart is greater than equal to the entered value.

Properties

This section describes the concepts of properties.

Overview

Generally, properties are used to extend products with miscellaneous data fields. Properties are attached to products with the help of *property groups*.

In particular properties are used to create *products with variants*, *configurable products* and *filters* or just to display generic information on a product. Additionally they can also be used to ask the shop customers to enter information for a product which is about to be ordered.

Types

There are three types of properties, which are described in detail in the following sections.

Text Field

The property's value are plain text without any logic behind it.

Float Field

The property value must be a float.

Step Type

The step type is used for *filtering*. It can be chose from three different methods how the steps are created.

Automatic The filter steps are calculate automatically. The base are the entered values of all products of a category for this property.

Fixed steps On base of the entered number LFS builds automatically all steps from the minimal to the maximum value of all entered values on the products of a category for this property.

Manual steps All steps, which are supposed to be displayed are entered manually.

Validators

Validators are used to validate customer inputs into property fields.

Decimal places The amount of decimal places which are allowed.

Min The minimum value of the field.

Max The maximum value of the field.

Step The valid steps from minimum to maximum value.

Example

- Min: 1
- Max: 3
- Steps: 0.5

Valid values which a shop customer could input into this field are: 1, 1.5, 2, 2.5 and 3.

Select Field

The property is displayed as a select box.

Select Field Properties

Display Price If this is checked the price of the options is displayed beside the name within the select box.

Add Price The price of the selected option of the property is added to the total price of a *Configurable Product*.

Select Field Options

Every single option for a `Select Field Property` has to be entered. These have following information:

- Position
- Name
- Price (optional)

The options are ordered by position, lower numbers are displayed first. The names are displayed within several selection fields. if `Add Price` is selected, the price is used to calculate the total price of a *Configurable Product*.

See Also

- *Properties Management Interface*
- *Properties within the Product Management Interface*
- *How To Add a Product with Filters*
- *How To Add a Product with Variants*
- *Local properties*

Local Properties

This section describes the concept of local properties.

Overview

Local properties are added to single products. They are a convenient way to create single variants for a *product with variants*. In contrary to *global properties* they can not be used to create *configurable products*, create filters or to display several informations on a product.

See Also

- *Global Properties*
- *Product with Variants*
- *Configurable Products*

Taxes

This section describes how LFS calculates taxes.

Note: If you don't have different taxes based on customers you can safely ignore the rest of this section.

Overview

There are two types of taxes: product taxes and customer taxes.

`Product taxes` define the default tax for a product and are used to calculate the default net or gross price of a product. Which one of these depends on the type of price you have entered for your product. Product taxes are assigned directly on the product.

`Customer taxes` define the tax for a specific customer respective for the country in which the goods are to be delivered. These taxes are managed centrally and selected automatically by LFS for the calculation.

Calculating Taxes

Following we'll explain how LFS calculates taxes.

Example 1

Assuming that the entered price of a product is inclusive tax, i.e. the entered price is the gross price of the product or in other words the selected `Price calculator` of the product is `Price Includes Tax`.

LFS will calculate the net price of the product on base of the entered `Product tax`, the tax you have assigned to the product.

LFS will then calculate the product's gross price for a customer on base of the customer tax which you have entered for the country of the customer's shipping address. If there is no customer tax for the customer LFS falls back to the product tax in order to calculate the gross price.

Example 2

Now let's imagine you have entered the price of the product exclusive tax, i.e. the net price of the product, or in other words the selected `Price calculator` of the product is `Price Excludes tax`.

LFS will just take the entered price for the product and calculates the product's gross price on base of the customer tax which you have entered for the country of the customer's shipping address. If there is no customer tax for the customer LFS falls back to the product tax in order to calculate the gross price.

Conclusion

In both examples you see that the product tax is used when no customer tax is found. This means if you don't have the need for different taxes based on customers you can safely ignore them at all and just assign your default taxes directly to the products.

See Also

- [*Manage product taxes*](#)
- [*Manage customer taxes*](#)

Static Blocks

This section describes the concept of static blocks.

Overview

A static block is a piece of information which are managed on a central place within LFS and can be reused from several content objects like the front page, [*categories*](#) or [*products*](#). It consists out of an arbitrary HTML text and an arbitrary amount of attachments which can be provided for download.

An example is a description, image or video than should be displayed on several products or categories.

See Also

- [*Static Blocks Management Interface*](#)

Portlets

This section describes the concept of portlets in general and the default portlets in detail.

Overview

A Portlet is a piece of arbitrary information which can be assigned to every object, like shop, products, categories and pages. They are displayed in slots. A slot can have an arbitrary amount of Portlets. LFS ships with a left and a right slot and several default portlets. By default portlets are inherited from parent objects but it is also possible to block parent portlets per slot.

The inheritance path for categories and products is:

```
shop > category > sub category > product
```

The inheritance path for pages is:

```
shop > page root > page
```

Default Portlets

This section describes all default portlets of LFS with their particular settings and properties.

Average Rating

This portlet displays the average rating of a product.

Title The title of the portlet.

Cart

This portlet displays the cart of the shop.

Title The title of the portlet.

Categories Portlet

This portlet displays the category tree, which is the essential navigation for the shop.

Title The title of the portlet.

Start Level The starting level of categories. If this is 2 the top level categories are not displayed within the portlet. This can be useful if you want to display them in the horizontal menu.

Expand Level Expands the categories up to this level. If this is 0 only the current category will be expanded. If this is 1 all top level categories are expanded, etc.

Note: top level categories have level 1, their sub categories have Level 2, etc.

Delivery Time

This portlet displays the delivery time of a product.

Title The title of the portlet.

Featured Products

This portlet displays products, which are selected within *Marketing / Featured*

Title The title of the portlet.

Limit Only the given amount of products are displayed.

Use current category If this is checked only the featured product of the current category are displayed.

Slideshow If this is checked the products are displayed with a slideshow, i.e. a single product which is exchanged automatically). If this is unchecked all products are display at once.

Filter

This portlet displays a filter portlet for a category.

Title The title of the portlet.

Show Product Filters: If this is checked product filters are displayed. To make this work properly the products assigned to the category must filterable properties and there must be values assigned to them.

Show price filters: If this is checked price filters are displayed (which are automatically calculated).

For Sale

This portlet displays products which are for sale.

Title The title of the portlet.

Limit Only the given amount of products are displayed.

Use current category If this is checked only the featured product of the current category are displayed.

Slideshow If this is checked the products are displayed via a slideshow, i.e. only one product at once which is exchanged automatically. If this is unchecked all products are display as a list.

Latest Products

Display the products that were recently added to the shop

Title The title of the portlet.

Limit Only the given amount of products are displayed.

Use current category If this is checked only the latest products of the current category are displayed.

Slideshow If this is checked the products are displayed with a slideshow, i.e. a single product which is exchanged automatically). If this is unchecked all products are display at once.

Pages

This portlet displays information pages.

Title The title of the portlet.

Recent Products

Display the recent visited products.

Title The title of the portlet.

Related Products

This portlet displays related products of a product.

Title The title of the portlet.

Text

This portlet displays arbitrary HTML.

Title The title of the portlet.

Text The HTML code which is supposed to be displayed.

Top Seller

This portlet displays the top seller of the shop.

Title The title of the portlet.

Limit Only the given amount of products are displayed.

See Also

- *Shop Preferences*
- *Categories Management Interface*
- *Products Management Interface*

Actions

This section describes the concepts of actions.

Overview

Actions are used to place links on several locations within the shop. They belong to exactly one action group and the action group decides the location the actions are displayed within the shop. LFS ships with two action groups, the tabs, which constitutes the horizontal menu bar and the footer. Developers can easily *add more action groups*.

Examples

Actions enables the shop owner to link to arbitrary categories, products, information pages or even external web sites.

See Also

- *Actions Management Interface*
- *How to add own action groups*

Delivery Times

This section describes the concepts of delivery times.

Overview

The delivery times of single products and the cart is calculated automatically by LFS. Delivery times are *managed centrally* and *are assigned to LFS' shipping methods*, that means they are generally dependent on the first valid or the selected shipping method of a customer, if this is not explicitly overwritten for a product (see below).

Products

To get the delivery time for a single product (to display it within the product page), LFS calculates the first valid shipping method for the product and the customer (all *criteria* are true) and takes its assigned delivery time. It's also possible to override this mechanism for single products with a *manually delivery time*.

Cart

To identify the delivery time for the total cart (to display it within the cart and the checkout page), LFS takes the shipping method the customer has currently selected and calculates on base of that the maximum delivery time of all products within the cart. The result can different from the selected shipping method as also in this case the manual delivery times of products are taken into account. Additionally the default delivery time is used if the selected shipping method for are product within the cart is not valid.

Miscellaneous

Additionally the *internal delivery time* (shop owner orders product) can be added to the delivery time for the customer.

See also

- *Delivery Times Management*
- *Shipping Method Management Interface*
- *Product Management Interface - Stock Data*
- *Criteria concepts*

Shipping Methods

This section describes the concept of shipping methods.

Overview

You can add as many *shipping methods* as you want.

All *valid* shipping methods are displayed for selection to the shop customer. A shipping method is valid if all *criteria* of the shipping method are true.

Shipping methods can have many prices which are, as the payment method itself, also dependent on criteria. The first price which is valid - all of its criteria are true - is the current price for the shipping method. If no price is valid the default price is taken (from the `Data` tab).

Shipping methods have also a *delivery time* which decides how long the delivery will take if a certain shipping method is selected by the customer.

See also

- *Shipping Method Management Interface*
- *Delivery Time Concepts*
- *Criteria Concepts*

Payment Methods

This section describes the concept of payment methods.

Overview

You can add as many payment methods as you want.

All valid payment methods are displayed for selection to the shop customer. A payment method is valid if all *criteria* of the payment method are true.

Payment methods can have many prices which are, as the payment method itself, also dependent of criteria. The first price which is valid - all of its criteria are true - is the current price for the payment method. If no price is valid the default price is taken (from the `Data` tab).

See Also

- *Payment Methods Management Interface*
- *How to add an own payment processor*
- *Criteria Concepts*

Marketing

This section describes the marketing features of LFS.

Accessories

Accessories are products which are supposed to be sold together with a product, like shingles to a summerhouse. They are not bidirectional but they need to be entered on each product itself.

Accessories are displayed within the `product view` and could be added to the cart alongside with the product. They are also displayed within the `added to cart view` (the view is displayed after a shop customer has added a product to the cart) in order to offer them to be also added to the cart.

Discounts

Discounts are a possibility to give a price reduction to customers. They can be absolute or percentaged based on the total cart price. Discounts can be restricted by *criteria* and they are only given if it meets all criteria. The shop owner can add an arbitrary amount of discounts.

Featured Products

Featured products are several product which are supposed to be exposed to the customer for any reason. They are *management manually* and can be displayed to the customer via the *featured product portlet*.

Rating Mails

Shop owners can send *rating mails* to shop customers in order to ask them to rate the products they bought. See also *review concepts* for more information.

Related Products

Related products are products which are somehow related to a product. They are not bidirectional related but they need to be assigned to every single product.

Related products can be displayed within the *Related Products Portlet*.

Top seller

Top seller are best sold products of the shop. They are calculated automatically but can be also *manipulated manually*. They are displayed within the *top seller portlet*

Vouchers

Vouchers are another possibility to give a price reduction for customers. The price reduction can be absolute or percentaged based on the total cart price.

They are generic character strings which can be distributed to customers. If a valid voucher string is entered within the cart or as part of the checkout process the customer gets the price reduction.

Vouchers can be limited by a start and an end date or a minimum cart price.

Vouchers expire when they have been used for a certain amount of times. A common way is to expire the voucher right after it has been used the first time.

The shop owner can add an arbitrary amount of vouchers. They can be of the same or of different types.

See also

- *Discounts Management Interface*
- *Products Management Interface*
- *Rating Mails Management Interface*
- *Review Concepts*
- *Vouchers Management Interface*

Reviews

This section describes the reviews of LFS.

Overview

Shop customers can review the products.

A review consists of a score of 1-5 stars and an arbitrary text. The average rating for a product is calculated automatically and displayed within the product view and the *average rating portlet*.

Logged in customers can review a product only once, anonymous customers just once in a session.

Shop administrators can influence the way reviews are handled, e.g. whether reviews are moderated or which fields are required. See *settings* for more.

Shop owners can send *rating mails* to shop customers in order to ask them to rate the products they bought.

See Also

- *Settings*
- *Rating Mails*

Page

This section describes the pages of LFS.

Overview

A page is a simple piece of HTML within the shop. It can be used to create landing pages or to display other information to shop customers like terms and conditions.

Pages are automatically displayed within the *pages portlet* or can be referred by *actions*.

Like other objects pages can have *portlets*.

See Also

- *Pages Management Interface*

Images

This section describes the concepts of images.

Overview

LFS provides two kinds of images: product images and global images.

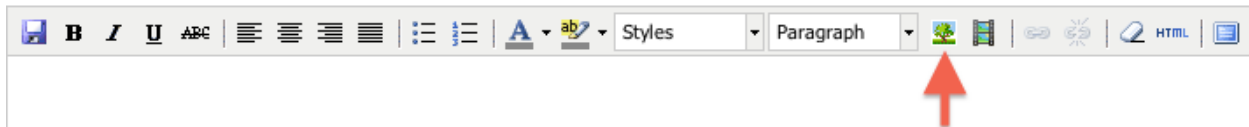
Product images

Product images managed within the *images tab* of a product and are attached to a single product. They are displayed automatically within the detail view of the product and the first image also within the category overview view.

Global images

Global images are managed on a *central place* and can be easily embedded within any WYSIWYG text field within LFS (e.g. within a *page*).

To do that one just have to click on the image icon of the WYSIWYG editor (see below), select the image in question, adapt the class and size of the image and click on the `insert image` button.



See also

- *Global image management*
- *Product images management*

Management Interface

Shop

This section describes the management interfaces of the Shop menu.

Actions

This section describes the management interface for *actions*.

Site Actions

Add Action Adds a new action.

Delete Action Deletes the currently displayed action.

View Shop This opens the front page of the shop in a pop-up window to quickly check the added actions without the need to leave the management interface.

Sort Actions In order to sort the actions, hold an action on the handle within the navigation on the left and drag and drop it to the position you want it to be.

Data

Active Only active actions are displayed to the shop user.

Title The title of the action. It is displayed to the shop user.

Link The URL to which the shop user is pointed when he clicks on the action.

Group The group the action belongs to. Select `Tabs` to display the action on the horizontal menu or `Footer` to display the action on the footer of the shop.

See also

- *General about actions*
- *How to add own action groups*

Delivery times

This section describes the management interface for delivery times.

Site Actions

Add Delivery Time Adds a new deliver time.

Delete Delivery Time Deletes the current displayed delivery time.

Data

Min The minimal delivery time.

Max The maximal delivery time.

Unit The unit of the delivery time, which is, hours, days, weeks or months.

Description A short description for internal uses only.

See also

- *General about delivery times*
- *Shipping methods*
- *Product stock data*

Manufacturers

This section describes the management interface for manufacturers.

Site Actions

Add Manufacturer Adds a new manufacturer.

Delete Manufacturer Deletes the current displayed manufacturer.

View Manufacturer Opens the Manufacturer in a pop-up window to quickly check the appearance of the Manufacturer without leaving the LMI.

Goto Manufacturer Leaves the LMI and goes to manufacturer view.

Tabs

Data

Name The name of the manufacturer

Slug The unique last part of the manufacturer's URL.

Short description The simplified description of the manufacturer. Displayed on the list of all manufacturers (/manufacturers).

Description The detailed description of the manufacturer. This is displayed on the detail view of the manufacturer.

Image Logo of the manufacturer.

View

Active formats If this check box is activated several formats for this manufacturer can be overwritten. Otherwise the formats are inherited from the shop.

Product cols / Product rows Amount of columns and rows which are used to display the products of the manufacturer. The amount of products which are displayed calculates by cols * rows. If there are more products than that the products are automatically paginated.

Note: This is only available if `Active Formats` is `True`.

SEO

This tab is used to optimize your pages for search engines. You can enter data for all meta data fields. However LFS provides some reasonable default values for all fields.

Meta title This is displayed within the meta title tag of the manufacturer's HTML tags. By default the name of the manufacturer is used.

Meta keywords This is displayed within the meta keywords tag of the manufacturer's HTML page. By default the short description of the manufacturer is used.

Meta description This is displayed within the meta description tag of the manufacturer's HTML page. By default the short description of the manufacturer is used.

Note: You can use several placeholders within these fields:

<name> The name of the manufacturer.

<short-description> The short description of the manufacturer (only within meta keywords and meta description field).

Products

In order to assign products to the manufacturer, select the check boxes of the corresponding products within the section `Selectable Products` and click on `Add To Manufacturer`.

Products Tree

Within this tab you can assign categories and products to the manufacturer.

To assign all products of a category to the manufacturer just check it. If you want just a sub category or single products of it, click on the category to expand the children.

Payment Methods

This section describes the management interface of payment methods.

Site Actions

Add Payment Method This add a new payment method

Delete Payment Method This deletes the currently displayed payment method.

Note: Default payment methods can't be deleted.

Tabs

Data

Name The name of the payment method, which is displayed to the shop customer (e.g. on the cart and the checkout page).

Description A short description of the payment method, which is displayed to the customer (e.g. on the checkout page).

Note A note of the payment method, which is displayed on the confirmation mail after the shop customer has been checked out.

Priority The first valid payment method with the highest priority (smaller number) method is displayed to the customer as default (if she hasn't selected one explicitly).

Image An image for the payment method, which is displayed to the shop customer.

Tax The included tax of the payment method's price

Price The default price of the payment method. This can be overwritten within the price tab (see below).

Module The dotted name of the external package which processes the payment (this is for developers only).

Type The type of the payment method. Dependent on that additional fields for input (within the checkout process) will be displayed. There are three types:

- Plain No additional fields are displayed.
- Bank Fields for a bank account are displayed.
- Credit Card Fields for a credit card are displayed.

Criteria

Here you can add criteria for the payment method. The payment method is only available for shop customers if all criteria are true.

Please see [How to manage payment methods](#) to see how to add criteria.

Prices

Here you can add additional prices for the payment method based on criteria. If prices are given the first price which meets all criteria is taken for the payment method. If no prices are given, the default price of the `Data` tab is taken.

Please see [How to manage payment methods](#) to see how to add prices.

See Also

- [General about payment method](#)
- [How to manage payment methods](#)
- [General about criteria](#)

Preferences

This section describes the preferences management interface.

Shop

General

Name The name of the shop. It will be used as prefix of all pages of the shop and is therefore very important for the appearance within SERPs.

Shop owner The shop owner. This can be used on several places within the shop, e.g. within the thank-you page.

E-Mails

From e-mail address This e-mail address will be used as sender for all outgoing mails.

Notification email addresses This e-mail addresses will be used as receiver for all incoming mails.

Google

Google Analytics ID The unique Google Analytics ID of the shop.

Google Analytics Site Tracking If checked the shop is tracked with Google Analytics.

Google Analytics E-Commerce Tracking If checked the shop is tracked with Google Analytics E-Commerce feature.

Checkout

Checkout type There are three checkout types:

Anonymous and Authenticated The customer can decide whether he registers and login to the shop or checkout without registration.

Anonymous only Registration is not available

Anonymous only All customers have to register and login to the shop.

Confirm TOC If checked the customers have to confirm the terms of contract of the shop in order to checkout.

Content

Description The description of the shop. This is displayed within the front page of the shop.

Static block The static block is displayed on top of the front page of the shop. See *Static Block* for more.

Image The image of the shop.

Default Values

Price calculator

Price calculator The default price calculator of the shop. If a product doesn't have an explicit price calculator this one is used.

Delivery time

Delivery time The default delivery time, which is used if no delivery time can be calculated for a product.

Format

Product cols Default value for amount of displayed columns within a category if the category displays products.

Product rows Default value for amount of displayed rows within a category if the category displays products.

Category cols Default value for amount of displayed columns within a category if the category displays sub categories.

Countries

Invoice countries Available countries within the invoice address.

Shipping countries Available countries within the shipping address.

Default shipping country The default shipping country within addresses. This country will be used to calculate shipping price if the shop customer doesn't have select a country yet.

Localization

Default shop locale This locale of the shop. This is used for localization of numbers and currencies, etc. You can find more information here: <http://en.wikipedia.org/wiki/Locale>

Use international currency codes If checked the international currency code of the currency is used, e.g. USD or EUR.

Order Numbers

Last order number This field stores the integer part of the last given order number, which is the base for the next given order number.

Format This field stores the format of the order number. The integer part which is stored in `Last order number` can be formatted with Python's string formatting operators, e.g.:

```
DOE-%04d-2012 will return DOE-0001-2012
```

Whereas `%04d` represents the integer part of the order number, which is stored in `Last order number`.

SEO

This tab is used to optimize the start page for search engines. One can enter data for all usual HTML meta data fields.

Meta title This is displayed within the `meta title` tag of the start page. By default the `Name` field of the `Shop` tab is used (see above).

Meta keywords This is displayed within the `meta keywords` tag of the start page.

Meta description This is displayed within the `meta description` tag of the start page.

Note: Following placeholder can be used within these fields:

`<name>` The name of the product.

Portlets

This tab is used to assign Portlets to the shop.

Slots Here you can see all directly assigned portlets to the shop. In order to edit a portlet click on row of the portlet. In order to delete a portlet click on the red cross beside the portlet. You can also change the position of the portlets by clicking on the up and down arrows beside the portlets.

Add new Portlet In order to add a portlet to the shop select the type of portlet and click on `Add portlet`.

Shipping methods

This section describes the management interface of shipping methods.

Site Actions

Add Shipping Method Adds a new shipping method.

Delete Shipping Method Deletes the currently displayed shipping method

Tabs

Data

Name The name of the shipping method, which is displayed to the shop customer (e.g. on the cart and the checkout page).

Description A short description of the shipping method, which is displayed to the customer (e.g. on the checkout page).

Note A note of the shipping method, which is displayed on the confirmation mail after the shop customer has been checked out.

Priority The first valid shipping method with the highest priority (smaller number) method is displayed to the customer as default (if she hasn't selected one explicitly).

Image An image for the shipping method, which is displayed to the shop customer (On the checkout page).

Tax The included tax of the shipping method's price.

Price The default price of the shipping method. This can be overwritten within the price tab (see below).

Delivery time The delivery time of the shipping method. See [Delivery times](#).

Criteria

Here you can add criteria for the shipping method. The shipping method is only available for shop customers if all criteria are true.

Please see [How to manage shipping methods](#) to see how to add criteria.

Prices

Here you can add additional prices for the shipping method based on criteria. If prices are given the first price which meets all criteria is taken for the shipping method. If no prices are given, the default price of the Data tab is taken.

Please see [How to manage shipping methods](#) to see how to add prices.

See also

- [General about shipping methods](#)
- [How to manage shipping methods](#)
- [General about criteria](#)

Product Taxes

This section describes the management interface for product taxes.

Site Actions

Add Tax Adds a new product tax.

Delete Tax Deletes the current product tax.

Data

Rate The tax rate in percent.

Description A short description for internal uses only.

See also

- *General about taxes*
- *Manage Customer Taxes*

Customer Taxes

This section describes the management interface for customer taxes.

Site Actions

Add Customer Tax Adds a new customer tax.

Delete Customer Tax Deletes the currently displayed customer product tax.

Data

Rate The tax rate in percent.

Countries The countries for which the customer tax is valid.

Note: You can add just one tax rate per country.

See also

- *Taxes in general*
- *Manage product taxes*

Images

This section describes the management interface for *global images*.

Data

Add images Click on the `Select images` button and select as many images as you want within your browsers pop-up window. You can use shift click to select a range of images at once and ctrl (cmd for apple users) click to select more images. Now click on open to start the upload process. You will now see a progress bar meanwhile your images are being uploaded.

Delete Images To delete images select the check boxes above all images you want to delete and click the `Delete` button. Alternative you can click on the `Toggle selection` button in order to inverse the current selection.

See also

- *Images concept*

Catalog

This section describes the management interfaces of the `Catalog` menu.

Categories

This section describes the category management interface.

Site Actions

Add Top Category Adds a category to the root of the catalog.

Add Category Adds a sub category to the current displayed category.

Delete Category Deletes the current displayed category.

View Category Opens the category in a pop-up window to quickly check the appearance of the category without leaving the LMI.

Goto Category Leaves the LMI and goes to customer view of the current displayed category.

Move Categories

In order to move categories, hold it on the handle left beside the category name and drag and drop it to the new place.

Tabs

Data

Exclude from navigation If checked the category is not displayed within the *categories portlet*.

Name The name of the category. This is displayed on several places, e.g: within category overviews and on top of the category detail view.

Slug The unique last part of the category's URL.

Description The detailed description of the category. This is displayed on the detail view of the category.

Image The image of the category. This can be displayed within category overviews and the category detail view.

Static block An optional *static block* which displayed on top of the category detail view.

View

Category template The selected template of the category decides whether the sub categories or the products are displayed. It is also possible to provide more templates to display single categories individually.

Show all products

If this check box is activated also the products of the sub categories are displayed. Otherwise only the direct products are displayed.

Note: This is only available when a the products of the category are displayed. (see Category Template).

Active formats If this check box is activated several formats for this category can be overwritten. Otherwise the formats are inherited from the parent object, which is either the parent category or the shop.

Category cols Amount of columns which are used to display the sub categories. Always all direct categories of the category are displayed.

Note: This is only available if active formats is True and a category template is selected.

Product cols / Product rows Amount of columns and rows which are used to display the products of the category. The amount of products which are displayed calculates by cols * rows. If there are more products than that the products are automatically paginated.

Note: This is only available if Active Formats is True and a product template is selected.

Products

This tab is used to assign and remove products to the category.

Add Products

In order to assign products to the category, select the check boxes of the corresponding products within the section `Selectable Products` and click on `Add To Category`.

Remove Products

In order to remove products from the category, select the check boxes beside the corresponding products within the section `Selected Products` and click on `Remove From Category`.

Filter

In order to make it easier to find products you can filter them by name, SKU and category. For that enter on top of the according section the name or the SKU into the text box and select the category out of the select box.

SEO

This tab is used to optimize your pages for search engines. You can enter data for all meta data fields. However LFS provides some reasonable default values for all fields.

Meta title This is displayed within the meta title tag of the category's HTML tags. By default the name of the category is used.

Meta keywords This is displayed within the meta keywords tag of the category's HTML page. By default the short description of the category is used.

Meta description This is displayed within the meta description tag of the category's HTML page. By default the short description of the category is used.

Note: You can use several placeholders within these fields:

<name> The name of the category.

<short-description> The short description of the category (only within meta keywords and meta description field).

Portlets

This tab is used to assign portlets to the category.

Blocked parent slots By default portlets are inherited from the parent category. To block portlets check the regarding slots and click on the `Save blocked parent slots` button.

Slots Here you can see all directly assigned portlets to the category. In order to edit a portlet click on row of the portlet. In order to delete a portlet click on the red cross beside the portlet. You can also change the position of the portlets by clicking on the up and down arrows beside the portlets.

Add new Portlet In order to add a portlet to the category select the type of portlet and click on `Add portlet`.

See also

- *Categories in general*
- *Portlets in general*

Products

This section describes the product management interface.

Site Actions

Add product Adds a product to the catalog.

Delete product Deletes the current displayed product.

View product Opens the product in a pop-up window to quickly check the appearance of the product without leaving the LMI.

Goto product Leaves the LMI and goes to customer view of the current displayed product.

Product type Select box to change the *type of the product*.

Tabs

Data

Active Only active products are displayed to the customer, can be found, bought etc.

Name The name of the product. This is displayed on the title of the product page, within overviews and as part of the HTML meta title tag.

Slug The last part of the product's URL. This must be unique.

Redirect to If the product is not active and this field is filled the user is redirected to the given URL. This might be useful if your product has been indexed by search engines but is not available any more. Note: super users are not redirected.

SKU Your unique product id.

Manufacturer Product *manufacturer*. If you have defined more than 20 manufacturers then this field is shown as auto complete (start typing to see hints).

SKU manufacturer The unique product id of the manufacturer (external SKU).

Price The price of the product. Whether this price is considered net or gross depends on the selected price calculator for a product (see below).

Price unit The price unit of the product. This is displayed after the price of the product within the product detail view and the category products view.

Tax The tax rate of the product. Whether this is included or excluded depends on the selected price calculator for a product (see below).

Price Calculation

Warning: This is experimental feature. Use it with care and take notice that these could be replaced with something different in future.

Note: This is only available for Configurable Products.

If the check box is checked the input field could contain a formula to calculate the price of a Configurable Product. In this case one can refer to the values of a property in order to calculate the price.

Generally the formula could contain any valid Python expression which is able to be evaluated with `eval` with two additional tokens.

If an error occurs the default price of the product is taken.

Available tokens:

```
product(<attribute>)
    Refers to the attribute of the current product

property(<id>)
    Refers to the value of the property with the given <id>
```

Example 1:

```
property(15) * product(price)
```

Which means: take the entered value of the property with the id 15 and multiply it with the product's price.

Example 2:

```
product(price) * property(54) * property(55) + property(56)
```

Which means: multiply the product's price with the values of the properties with ids 54, 55 and add the value of the property with the id 56.

Price calculator Determines how the product price is calculated using the product price and tax stored in the database. If you leave this field blank, your pricing calculator will default to the shop *price calculator*.

LFS ships with two pricing calculator methods: `Price Includes Tax`, which means the product price in the database includes tax and `Price Excludes Tax`, which means the product price in the database excludes tax.

For sale If the check box is activated the entered for sale price is active. On all views the standard price is displayed stroked and the for sale price is displayed emphasized.

For variants following is true:

Standard Inherits the `for sale` state of the base article.

Yes Variant is for sale.

No Variant is not for sale.

Quantity field unit This is displayed before the quantity field of the product within the product detail view and after the product amount within cart and order items.

Type of quantity field There are three types of quantity fields: `Integer`, which means the quantity must be an integer and all decimal places are ignored. `Decimal 0.1`, which means the quantity must be a decimal number with one place and more decimal places are ignored. `Decimal 0.01`, which means the quantity must be a decimal number with two places and more decimal places are ignored.

Active base price If this is activated the base price of the product is displayed within product detail view and category products view.

For variants following is true:

Standard Inherits the activate base price from the base article. Values for `base price unit` and `base price amount` are taken from the base article.

Yes Base price is activated. Values for `base price unit` and `base price amount` are taken from the variant.

No Base price is deactivated.

Base price unit This unit is displayed after the base price of the product.

Base price amount The amount, which is used to calculate the base price of the product. The base price of the product is:

$$\text{base price} = \text{price} / \text{base price amount}$$

Short description A short description of the product. This is displayed within overviews like categories or the search result page.

Description The detailed description of the product. This is displayed within the product page.

Static block An optional static block which displayed on top of the product view.

Product template The selected product template decides how the content of the product is structured.

Categories

Within this tab you can assign categories to the product. To do that just select all categories the product should be a part of and click on `Save Categories`.

Note: You can also *assign products to categories*.

Images

Within this tab you can add images to the product.

Images are displayed on the details view of the product. The first image is the default image of the product and is also displayed on overviews like the category detail view or search results view.

Add images Click on the `Select images` button and select as many images as you want within your browsers pop-up window. You can use shift click to select a range of images at once and ctrl (cmd for apple users) click to select more images. Now click on open to start the upload process. You will now see a progress bar meanwhile your images are being uploaded.

Update images To update the images just change the Title and the position of all products you want to change and click on the `Update` button.

Move images To move images just click on the up or down arrow beside the image.

Delete Images To delete images select the check boxes beside all images you want to delete and click the `Delete` button.

Attachments

Within this tab you can add attachments to the product. They are displayed for download on the detail view of the product.

Add Attachments Click on the `Select files` button and select as many attachments as you want within your browsers pop-up window. You can use shift click to select a range of images at once and ctrl (cmd for apple users) click to select more images. Click on select to start the upload process. You will now see a progress indicator meanwhile your images are being uploaded.

Update attachments To update the images just change the Title and/or the position of all products you want to change and click on the `Update` button.

Move attachments To move attachments you just click on the up or down arrows beside the attachment.

Delete attachments To delete attachments select the check boxes beside all images you want to delete and click the `Delete` button.

Variants

Within this tab you can manage the variants of a `Product with Variants`.

Note: This is only displayed for `Products with Variants`.

Property Groups

Select all property groups which are supposed to be used to create variants. After you have selected the property groups you want, you will notice that the properties of the groups are provided to create variants within the `Variants` section below.

Note: Only properties with select fields will be taken into account.

Local Properties

`Local Properties` can be used to create variants without using `Property Groups`. To add properties click on the stencil and add properties and property options. After you add local properties you will note that these are provided to create variants within the `Variants` section below.

Note: Local properties can not be used for filtering.

Variants

Within this section single variants of the `Product with Variants` are managed.

Add Variants To add variants to the `Product with Variants`, select the options combination you want to add and click on the `Add Variant(s)` button. If you select all combinations of this property and its options will be created automatically.

You can pre-fill several fields of the new variants. All fields can be changed later.

Slug The slugs of the variants will be pre-filled with the slug of the base product, plus the slug you provide, plus all options of the properties for which are selected for the variant.

Name The name of the variants will be pre-filled with the name you provide.

Price The price of the variants will be pre-filled with the price you provide.

Edit Variants There are several fields of the variants which you can edit directly on this section. All others can be edit on the variant detail view.

Position The position of the variant within the list view.

Active If checked the variant is displayed.

URL The URL of the variant

SKU The SKU of th variant. This is only taken if the check box on the left is checked. Otherwise the SKU of the base product is taken.

Name The name of th variant. This is only taken if the check box on the left is checked. Otherwise the name of the base product is taken.

Price The price of th variant. This is only taken if the check box on the left is checked. Otherwise the price of the base product is taken.

Default The default variant, which pre-selected when the product is displayed.

To save changed variants click on the `Save` button.

Delete Variants To delete variants select all check boxes of the variants you want to delete and click on the `Delete` button.

Category Variant

The category variant determines which variant is displayed within the category products view.

Default Displays the above selected default variant

Cheapest Price Displays the variant with the cheapest price

Cheapest Base Price Displays the variant with the cheapest base price

Cheapest Prices Displays the variant with the cheapest price and cheapest base price

Explicit Variant Displays selected variant (all variants are provided for selection by name and position).

Display Type

The display type determines how variants are displayed within the product detail view

List All existing variants are displayed within a list.

Select All properties are displayed as select boxes with the property options as options.

Note: If the customer selects a combination, which doesn't exist he will get a message which says so.

Accessories

Within this tab you can manage the *accessories* of a product.

Add accessories

Within the `Selectable Products` section select all check box beside the product you want to add as accessory to the product and click on `Add To Accessories`.

Note: You can filter the selectable products by name and category with the input fields on top of the `Selectable Products` section.

Update accessories

Within the `Selected Products` section change the values you want and click on `Save accessories`.

Position The position within the product. Lower numbers are displayed first.

Quantity The entered quantity is displayed next to the accessory. The shop customer can only add the given quantity to the cart.

Remove accessories

Within the `Selected Products` section select all check boxes beside the products you want to remove from the product and click on `Remove From Accessories`.

Related products

Within this tab you can manage the *related products* of a product.

Add related products

Within the `Selectable Products` section select all check box beside the product you want to add as related products to the product and click on `Add To Related Products`.

Note: You can filter the selectable products by name and category with the input fields on top of the `Selectable Products` section.

Remove related products

Within the `Selected Products` section select all check boxes beside the products you want to remove from the product and click on `Remove From Related Prouducts`.

Stock

Within this tab you can manage all stock related information of the product, like the dimension, stock amount and delivery dates.

Dimension

The values of the product which are considered shipping relevant, i.e. the product within its package.

Weight The weight of the product.

Height The height of the product.

Width The width of the product.

Length The length of the product.

Stock data

Deliverable If this is deactivated the product is not deliverable at all. The shop customer sees the product but he is not able to add the product to the cart.

Manual delivery time By default the delivery time is calculated automatically by the currently valid shipping method for a product. With this field the shop owner can overwrite this behavior and can put in a manual delivery time.

Manage stock amount If this is checked the stock amount will be decreased when the product has been bought. Additionally the maximum amount which can be bought is the number in `Stock amount` (see below).

Stock amount The available amount of the product in stock.

Order time Duration from ordering the product to being in stock again (when it is out of stock).

Ordered at The date when the **shop owner** has ordered the product.

Note: If `Order time` and `Order at` is given the total `delivery time` is calculated based on this two fields and the default `Delivery time`.

Packing

Active packing If this is checked the product can only be sold in packings.

For variants following is true:

Standard Inherits the packing state from the base article. Values for `packing amount` and `packing unit` are taken from the base article.

Yes Packing is activated. Values for `packing amount` and `packing unit` are taken from the variant.

No Packing is deactivated.

Packing amount Amount of products per packing.

Packing unit: The unit of the packing. This is displayed after the packing amount.

SEO

This tab is used to optimize the product for search engines. One can enter data for all usual HTML meta data fields. However LFS provides some reasonable default values for all fields.

Meta title This is displayed within the `meta title` tag of the product's detail view. By default the name of the product is used.

Meta keywords This is displayed within the `meta keywords` tag of the product's detail view. By default the short description of the product is used.

Meta description This is displayed within the `meta description` tag of the product's detail view. By default the short description of the product is used.

Note: Following placeholders can be used within these fields:

<name> The name of the product.

<short-description> The short description of the product (only within meta keywords and meta description field).

Portlets

This tab is used to assign portlets to the product.

Blocked parent slots By default portlets are inherited from the current category. To block portlets check the regarding slots and click on the `Save blocked parent slots` button.

Slots Here you can see all directly assigned portlets to the product. In order to edit a portlet click on row of the portlet. In order to delete a portlet click on the red cross beside the portlet. You can also change the position of the portlets by clicking on the up and down arrows beside the portlets.

Add new portlet In order to add a portlet to the product select the type of portlet and click on `Add portlet`.

Properties

This tab is used to assign properties to the product (via property groups) and add values to them.

To do that select the `Property groups` you want to assign to the product and click on `Update property groups`. Then enter the values for the properties you want and click on `Update properties`.

Dependent on the kind of the property you can add values for the default value, the filter value and the displayed value.

See Also

- *Products in general*
- *Portlets in general*
- *Properties in general*

Products Overview

This section describes the products overview management interface.

Overview

Here you can bulk edit products.

Change Products

In order to delete products activate the check box on the left side of the products in question and click the `Delete` button. In order to modify products make the changes in the product fields and click the `Save` button.

Warning: Deletions or modifications can't be made undone.

Filtering

In order to display just a sub set of products you want to change you can use the filter section on top of the screen. A product must join all filters in order to be displayed, in other words the filter fields are joined by a logical AND. To

set the filters click on the `Submit` button. To reset the filters click on the red button on the most right site of the filter section.

Filter fields

Amount The amount of products which are displayed on one page. If there are more within the current filter settings you can skim through it with the help of the page navigation on the left side of the filter area.

Name/SKU Filters the products by given name or SKU.

Category Filters the products by given category.

Active Filters the products by given activity state.

For Sale Filters the products by the for sale state.

Sub type Filters the displayed products by the entered sub type.

Properties

This section describes the management interfaces of the `Properties` menu.

Property Groups

This section describes the management interface for `property groups`.

Site Actions

Add Property Group Adds a new property group.

Delete Property Group Deletes the current displayed property group.

Warning: Please note that all values which have been assigned to products and properties of this property group will get lost.

Tabs

Data

Name The unique name of the property group. This is used to assign property groups to products.

Properties

Within this tab you can manage properties for this property group.

Add Properties To add properties to the group select the check box beside the properties you want to add within the `Properties` section on the left side and click on `Add to Property Group`.

Update Properties To change the position of properties enter the new position of the properties and click on `Update properties`.

Remove Properties To remove properties from the group select the check boxes beside the properties you want to remove within the `Selected properties` section on the right side and click on `Remove from Property Group`.

Warning: Please note that all values which have been assigned to products of this group and the removed property will get lost.

Products

Within this tab you can assign products to the property group. All products will have the properties of this property group.

Note: The selectable products can be filtered with the text field (name) and select box (categories).

Add Products To add products to the group select the check box beside the products you want to add within the `Products` section on the left side and click on `Add to Property Group`.

Remove Products To remove products from the group select the check boxes beside the products you want to remove within the `Selected products` section on the right side and click on `Remove from Property Group`.

Warning: Please note that all values which have been assigned to properties of this group and this product will get lost.

Product values

In this tab you can assign values for every product / property pair within this group.

For that just enter the you want and click save values.

Note: You can also enter the values within the `Properties` tab of the product. *[See here for more](#)*.

See More

- *[Properties within Product Management Interface](#)*.

Properties

This section describes the management interface for properties.

Site Actions

Add Property Adds a new property.

Delete Property Deletes the current displayed property.

Warning: Please note that all values which have been assigned to products for this property will get lost.

Data

Position The position within the product.

Name The unique name of the property. This is displayed within the management interface to select the property for several purposes.

Title The title of the property. This is displayed to the shop customer.

For Variants: If this is checked the property will be used to create variants. Please be aware that only select fields can be used to create variants.

Filterable If this is checked the property will be displayed for filtering (within the filter portlet).

Display on product If this is checked the property will be displayed on the product.

Configurable If this is checked the property will be displayed for selection/input on a configurable product

Required If this is checked the property must be selected/entered on a configurable product. This is only displayed if `Configurable` is checked.

Unit The unit of the property, e.g. meter, liter or whatever you want. This is just a simple string without logic.

Field type The field type of the property. Based on the type of the field you need to provide additional data. See [Properties in general](#) for more information.

See also

- [Properties in general](#)
- [Property Groups Management Interface](#)

HTML

This section describes the management interfaces of the HTML menu.

Pages

This section describes the management interfaces of pages.

Site Actions

Add Page Adds a new page.

Delete Page Deletes the currently displayed page.

View Page Opens the page in a pop-up window to quickly check the appearance of the page without leaving the LMI.

Goto Page Leaves the LMI and redirects to the customer view of the current displayed page.

Sort Pages

In order to sort pages, take the handle on besides a page within the navigation on the left side and drag and drop it to the position you want the page to be.

Root

This is the root of all pages. You can add portlets to it which are inherited from all other pages.

Tabs

Data

Title The title of the page. This is displayed on top of the page as well as within the meta title tag.

Slug The unique last part of the URL to the page.

Short text The short text of the page. This is displayed within overviews.

Text The main text of the page.

Active If this is checked the page is active. Only active pages are displayed to the shop users.

Position Pages are ordered by position. Lower numbers come first.

File A file which can be uploaded. If a file has been uploaded a download link is automatically displayed at the bottom of the page.

SEO

This tab is used to optimize your pages for search engines. You can enter data for all usual HTML meta data fields. However LFS provides some reasonable default values for all fields.

Meta title This is displayed within the meta title tag of the category's HTML tags. By default the name of the product is used.

Meta keywords This is displayed within the meta keywords tag of the category's HTML page. By default the short description of the category is used.

Meta description This is displayed within the meta description tag of the category's HTML page. By default the short description of the category is used.

Note: You can use several placeholders within these fields:

<name> The name of the page.

<short-description> The description of the page (only within meta keywords and meta description field).

Portlets

This tab is used to assign portlets to the page.

Blocked parent slots By default portlets are inherited from the current category. To block portlets check the regarding slots and click on the `Save blocked parent slots` button.

Slots Here you can see all directly assigned portlets to the page. In order to edit a portlet click on row of the portlet. In order to delete a portlet click on the red cross beside the portlet. You can also change the position of the portlets by clicking on the up and down arrows beside the portlets.

Add new Portlet In order to add a portlet to the page select the type of portlet and click on `Add portlet`.

See Also

- *Pages in general*
- *Portlets in general*

Static Block

This section describes the management interface of `Static Blocks`.

Site actions

Add Static Block Adds a Static Block to the shop.

Delete product Deletes the current displayed Static Block.

View Static Block Opens the current displayed Static Block in a pop-up window to quickly check the appearance of the without leaving the LMI.

Tabs

Data

Name The name of the Static Block. The name is used to select a certain static block.

Display Files This must be checked in order to display the attached files.

HTML The HTML of the Static Block. This can be any valid HTML code.

Files

Within this tab you can add arbitrary files to the Static Block. These are provided for download if `Display Files` is checked (see above).

Add files Click on the `Select files` button and select as many files as you want within your browsers pop-up window. You can use shift click to select a range of files at once and ctrl (cmd for apple users) click to select more files. Now click on open to start the upload process. You will now see a progress bar meanwhile your files are being uploaded.

Update files To update the files just change the Title and/or the position of all filed you want to change and click on the `Update` button.

Move files To move files just enter the position of every file and click on the `Update` button.

Delete files To delete files select the check boxes beside all files you want to delete and click the `Delete` button.

See also

- *Static Blocks*

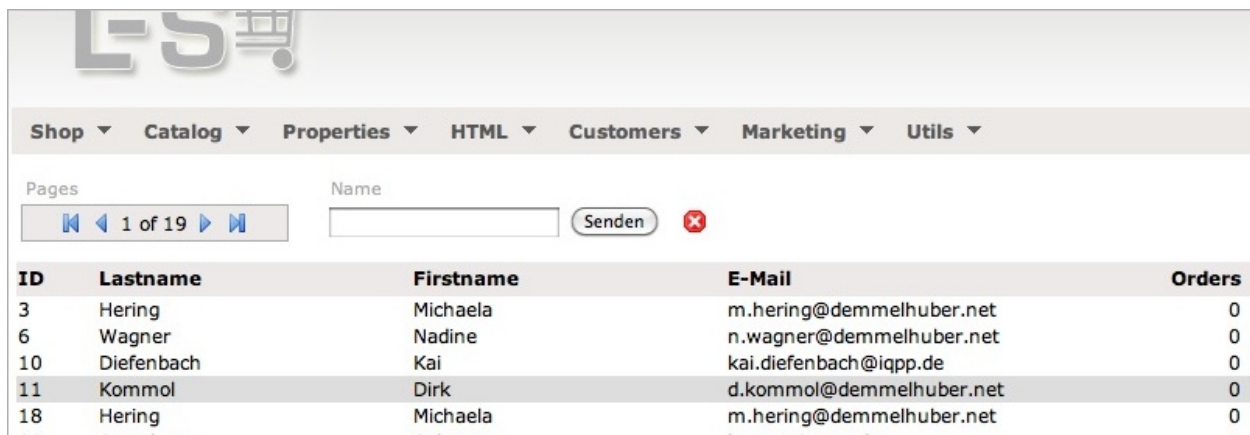
Customers

This section describes the management interfaces of the `Customers` menu.

Customers

Overview

Within the customers overview all customers are displayed.



ID	Lastname	Firstname	E-Mail	Orders
3	Hering	Michaela	m.hering@demmelhuber.net	0
6	Wagner	Nadine	n.wagner@demmelhuber.net	0
10	Diefenbach	Kai	kai.diefenbach@iqpp.de	0
11	Kommol	Dirk	d.kommol@demmelhuber.net	0
18	Hering	Michaela	m.hering@demmelhuber.net	0

Navigation / Filtering

With the page navigation you can navigate through all customers. To filter the displayed customers by name just fill in the name field and click `Submit`. To reset the filter just click on the red `X` button.

Sorting

To sort the displayed customers click on the columns title.

Details

To view a customer in detail just click on the row of the customer.

Details

Within the customer detail view you can see several information about a single customer: the id and the registration status, the current invoice address and shipping address, the current cart of the customer, if there is one. Click on the cart row to see it in detail. You can also check which orders the customer has submitted and the state of the them. Click on the order row to see it in detail.

Navigation: 1 of 19

Name:

Michaela Hering
Nadine Wagner
Kai Diefenbach
Dirk Kommol
Michaela Hering

[↑ Overview](#)

General

ID: 18
Registriert: No

Invoice address **Shipping address**

Michaela Hering (m.hering@demmelhuber.net)
Am gewerbegebiet 3
09661 hainichen
03720768362

Cart

ID	Modification date
21	

Navigation / Filtering

With the page navigation you can navigate through all customers. To filter the displayed customers by name just fill in the name field and click Submit. To reset the filter just click on the red X button.

Orders

Overview

Within the orders overview all orders are displayed.

Pages: 1 of 1

Start: End: Name: State:

Date	Name	State	Price	Message
Dec. 26, 2009, 2:20 p.m.	Kai Diefenbach	Submitted	5,36 EUR	No
Dec. 16, 2009, 4:26 p.m.	Kai Diefenbach	Submitted	13,46 EUR	No
Dec. 16, 2009, 4:24 p.m.	Kai Diefenbach	Submitted	15,44 EUR	No
Sept. 17, 2009, 12:47 p.m.	Kai Diefenbach	Submitted	9,10 EUR	Yes
Sept. 17, 2009, 9:39 a.m.	Kai Diefenbach	Submitted	2,20 EUR	No
Sept. 17, 2009, 9:14 a.m.	Kai Diefenbach	Submitted	0,01 EUR	No
Sept. 14, 2009, 1:13 p.m.	Kai Diefenbach	Submitted	0,01 EUR	No
Aug. 24, 2009, 9:31 a.m.	Kai Diefenbach	Closed	12,10 EUR	No
Aug. 6, 2009, 6:33 p.m.	Kai Diefenbach	Closed	2,20 EUR	No
Aug. 4, 2009, 10:56 a.m.	Kai Diefenbach	Closed	2,20 EUR	No
Aug. 4, 2009, 10:52 a.m.	Kai Diefenbach	Closed	2,20 EUR	No

Navigation / Filtering

You can filter and navigate through the displayed orders:

Navigation: 1 of 29

Start: End: Name: State:

- With the page navigation you can navigate through all orders.
- To filter the orders by name just fill in the name field and click `Submit`.
- To filter the orders by state just select the state and click `Submit`.
- To filter the orders by date just fill in `Start` and `End` (YYYY-MM-DD) and click `Submit`. For convenience you can click on the links left beside the start field. T = today, Y = yesterday, W = week.
- To reset the filter just click on the red X button.

Sorting

Note: Sorting is not available at the moment.

Details

- To view a order in detail just click on the order.

Details

Within the order detail view you can see several information about a single order:

- The general information: customer name, e-mail address and phone. The date, state and total amount of the order.
- Shipping and invoice address of the customer. The shipping and payment method of this order.
- The order items.
- On the left you can see all other orders. Just click on an order to see it in detail.

1 of 1

Start

End

Name

State

T Y W

diefenbach

All

Senden

X

↑ Overview

⊖ Delete order

✉ Resend order

State: Submitted

Change

General

Customer name:

Kai Diefenbach

Date:

Dec. 26, 2009, 2:20 p.m.

E-mail:

kai.diefenbach@iqpp.de

State:

Submitted (Dec. 26, 2009, 2:20 p.m.)

Phone:

0361 4711

Total:

5,36 EUR

Shipping address

Invoice address

Shipping method

Payment method

Hans Dampf

Kai Diefenbach

Dampf Str.

Klausener Str. 35

88888 Dampf Stadt

99099 Erfurt

Deutschland

Deutschland

0361 4711

+49 (361) 6636700

Paketversand

PayPal

SKU

Name

Amount

Price

Included VAT

Total

06-009-01-001-00006

Holzpuzzle

1

5,95 EUR

(0,95 EUR)

5,95 EUR

Voucher (UHGK)

1

- 0,59 EUR

- 0,59 EUR

Shipping (Paketversand)

1

0,00 EUR

0,00 EUR

Payment (PayPal)

1

0,00 EUR

0,00 EUR

5,36 EUR

Dec. 26, 2009, 2:20 p.m.

Dec. 16, 2009, 4:26 p.m.

Dec. 16, 2009, 4:24 p.m.

Sept. 17, 2009, 12:47 p.m.

Sept. 17, 2009, 9:39 a.m.

Sept. 17, 2009, 9:14 a.m.

Sept. 14, 2009, 1:13 p.m.

Aug. 24, 2009, 9:31 a.m.

Aug. 6, 2009, 6:33 p.m.

Aug. 4, 2009, 10:56 a.m.

Aug. 4, 2009, 10:52 a.m.

Aug. 2, 2009, 6:50 p.m.

July 25, 2009, 5:16 p.m.

Navigation / Filtering

You can filter and navigate through the displayed orders on the left:

- With the page navigation you can navigate through all orders.
- To filter the orders by date just fill in Start and End (YYYY-MM-DD) and click Submit. For convenience you can click on links beside the start field. T = today, Y = yesterday, W = week.
- To filter the orders by name just fill in the name field and click Submit.
- To filter the orders by state just select the state and click Submit.
- To reset the filter just click on the red X button.

Actions

Delete order To delete the current order just click on the `Delete order` button and answer the confirmation question with `Yes`.

Resend order You can resend the current displayed order to the customer. To do that click on the `Resend` button and answer the confirmation question with `Yes`.

Change state: To change the state of the current order select the desired state and click on the `Change` button.

Carts

Overview

Within the carts overview all carts are displayed.

Pages						
<div> <div>1 of 118</div> <div>T Y W</div> <div>Start</div> <div>End</div> <div>Senden</div> <div>X</div> </div>						
ID	Creation date	Modification date	User	Total	Items	Products
4316	Dec. 13, 2009, 11:48 a.m.	Feb. 20, 2010, 10:19 a.m.	Kai Diefenbach	1,00 EUR	1	Helo Sun Infrarotkat
4651	Feb. 9, 2010, 10:50 p.m.	Feb. 9, 2010, 10:54 p.m.	---	797,00 EUR	3	Jungle Gym SWING HOUSE Modul - M, S Grün
4650	Feb. 9, 2010, 8:24 p.m.	Feb. 9, 2010, 8:26 p.m.	---	333,98 EUR	3	Weber - Holzkohlekö Holzkohlegrill - One-
4649	Feb. 9, 2010, 8:11 p.m.	Feb. 9, 2010, 8:13 p.m.	---	0,00 EUR	0	
4648	Feb. 9, 2010, 7:15 p.m.	Feb. 9, 2010, 7:15 p.m.	---	219,00 EUR	1	Weber Gasgrill - Web
4647	Feb. 9, 2010, 3:05 p.m.	Feb. 9, 2010, 3:05 p.m.	---	1.254,00 EUR	4	Gartenhaus Superior 2 - für Gartenhäuser
4646	Feb. 9, 2010, 1:11 p.m.	Feb. 9, 2010, 1:11 p.m.	---	279,05 EUR	1	Spielturm Jungle Gyi
4644	Feb. 8, 2010, 9:58 p.m.	Feb. 8, 2010, 9:58 p.m.	---	519,00 EUR	1	Spielturm Jungle Gyi
						Spielturm Jungle Gyi

Navigation / Filtering

You can filter and navigate through the displayed carts:

Pages

1 of 118

Start

T Y W

End

Senden

✖

- With the page navigation you can navigate through all carts.
- To filter the carts by date just fill in Start and End (YYYY-MM-DD) and click `Submit`. For convenience you can click on the links left beside the start field. T = today, Y = yesterday, W = week.
- To reset the filter just click on the red X button.

Sorting

Note: Sorting is not available at the moment.

Details

- To view a cart in detail just click on the cart.

Details

Within the cart detail view you can see several information about a single cart:

- The general information: cart id, the customer (if there is one), creation and modification date, amount of items and the total amount.
- The cart items.
- On the left you can see all carts. Just click on a cart to see it in detail.

Dec. 13, 2009, 11:48 a.m.					
Feb. 9, 2010, 10:50 p.m.					
Feb. 9, 2010, 8:24 p.m.					
Feb. 9, 2010, 8:11 p.m.					
Feb. 9, 2010, 7:15 p.m.					
Feb. 9, 2010, 3:05 p.m.					
Feb. 9, 2010, 1:11 p.m.					
Feb. 8, 2010, 9:58 p.m.					
Feb. 8, 2010, 9:26 p.m.					
Feb. 8, 2010, 4:32 p.m.					
Feb. 8, 2010, 6:48 a.m.					
Feb. 7, 2010, 10:05 p.m.					
Feb. 7, 2010, 9:06 p.m.					
Feb. 7, 2010, 6:59 p.m.					
Feb. 7, 2010, 6:09 p.m.					
Feb. 7, 2010, 6:02 p.m.					
Feb. 7, 2010, 5:45 p.m.					
Feb. 7, 2010, 1:50 p.m.					
Feb. 6, 2010, 8:21 p.m.					
Feb. 6, 2010, 6:54 p.m.					
Jan. 18, 2010, 5 p.m.					
Feb. 6, 2010, 3:42 p.m.					
Feb. 6, 2010, 3:15 p.m.					
Feb. 6, 2010, 11:56 a.m.					
Feb. 6, 2010, 10:10 a.m.					
Feb. 4, 2010, 8:48 a.m.					
Feb. 5, 2010, 3:19 p.m.					

Overview

General

ID: 4650
Customer: ---
Creation date: Feb. 9, 2010, 8:24 p.m.
Modification date: Feb. 9, 2010, 8:26 p.m.
Items: 3
Total: 333,98 EUR

ID	Creation date	Product	Price	Amount	Total
8541	Feb. 9, 2010, 8:24 p.m.	Weber - Holzkohle Körbe Char-Basket	19,99 EUR	1	19,99 EUR
8542	Feb. 9, 2010, 8:24 p.m.	Weber - Anzündkamin Rapidfire	24,99 EUR	1	24,99 EUR
8543	Feb. 9, 2010, 8:26 p.m.	Weber Holzkohlegrill - One-Touch Gold Special Edition	289,00 EUR	1	289,00 EUR

Navigation / Filtering

You can filter and navigate through the displayed carts on the left:

Pages

1 of 118

Start

T Y W

End

Senden

×

- With the page navigation you can navigate through all carts.
- To filter the carts by date just fill in Start and End (YYYY-MM-DD) and click Submit. For convenience you can click on links beside the start field. T = today, Y = yesterday, W = week.
- To reset the filter just click on the red X button.

Reviews

Overview

Within the reviews overview all existing reviews are displayed.

Pages		Name	Active				
1 of 11				Daten absenden			
ID	Creation date	Name	E-Mail Product	Score	Comment	Active	
3	10.09.2008	---	---	Jungle Gym Rutsche - Wellenrutsche 2,40 m GS/TÜV	5.0	✓	✓
4	15.09.2008	---	---	Jungle Gym Rutsche - Wellenrutsche 2,40 m GS/TÜV	5.0	✗	✓
5	18.09.2008	---	---	Jungle Gym Rutsche - Wellenrutsche 2,40 m GS/TÜV	5.0	✗	✓
7	25.03.2009	---	---	Jungle Gym Rutsche - Wellenrutsche 2,40 m GS/TÜV	5.0	✓	✓
9	15.05.2009	---	---	Jungle Gym Rutsche - Wellenrutsche 2,40 m GS/TÜV	5.0	✓	✓
10	15.05.2009	---	---	Jungle Gym Rutsche - Wellenrutsche 2,40 m GS/TÜV	5.0	✓	✓
13	09.09.2008	---	---	Sitzgruppe HOT SPRINGS - Lake Sylva	5.0	✗	✓
14	09.09.2008	---	---	Klapptisch eckig - Lake Sylva	5.0	✓	✓
15	15.09.2008	---	---	Klapptisch eckig - Lake Sylva	5.0	✓	✓
16	24.03.2009	---	---	Klapptisch eckig - Lake Sylva	5.0	✓	✓
19	17.05.2009	---	---	Klapptisch eckig - Lake Sylva	4.0	✓	✓
20	21.09.2008	---	---	Jungle Gym - Schaukelsystem SWING	5.0	✓	✓
21	15.05.2009	---	---	Jungle Gym - Schaukelsystem SWING	5.0	✓	✓
22	15.05.2009	---	---	Jungle Gym - Schaukelsystem SWING	5.0	✓	✓
29	08.09.2008	---	---	Strickliesel	5.0	✗	✓

Navigation / Filtering

You can filter and navigate through the displayed reviews:

Pages

1 of 11

Name

Active

Daten absenden

×

- With the page navigation you can navigate through all customers.
- To filter the reviews by name just fill in the name field and click Submit.
- To filter the reviews by activity state just select the state and click Submit.
- To reset the filter just click on the red X button.

Sorting

- To sort the displayed reviews click on the columns title.

Details

- To view a review in detail just click on it.

Details

Within the review detail view you can see several information about a single review:

- The general information: the id, activity state, creation date, reviewer's name, reviewer's e-mail, customer's registration state, the reviewed state, the score.
- The comment of the review

1 of 11

Name
Active
Daten absenden

10.09.2008 - 08:11
15.09.2008 - 09:02
18.09.2008 - 14:54
25.03.2009 - 18:43
15.05.2009 - 18:26
15.05.2009 - 10:35
09.09.2008 - 06:35
09.09.2008 - 07:51
15.09.2008 - 06:05
24.03.2009 - 16:32
17.05.2009 - 17:43
21.09.2008 - 18:39
15.05.2009 - 07:57
15.05.2009 - 12:28
08.09.2008 - 09:21
15.05.2009 - 06:52
17.05.2009 - 17:41
15.05.2009 - 06:27
09.09.2008 - 19:22
10.09.2008 - 10:45

Overview
Delete review
Active: Yes
Change

General
ID: 7
Active: Yes
Creation date: 25.03.2009 - 18:43
Name: ---
E-Mail: ---
Registriert: No
Product: Jungle Gym Rutsche - Wellenrutsche 2,40 m GS/TÜV
Score: 5.0
Comment:
die leiferung hat sehr kurzfristig funktioniert.
das gewünschte produkt wurde richtig und in bester Qualität geliefert.

Navigation / Filtering

You can filter and navigate through the displayed reviews on the left:

Pages
1 of 11

Name
Active
Daten absenden

- With the page navigation you can navigate through all customers.
- To filter the reviews by name just fill in the name field and click Submit.
- To filter the reviews by activity state just select the state and click Submit.
- To reset the filter just click on the red X button.

Actions

Overview
Delete review
Active: Yes
Change

Delete review To delete the current review just click on the `Delete review` button and answer the confirmation question with `Yes`.

Change activity state: To change the activity state of the current review select the desired state and click on the `Change` button.

Note: Only active reviews are displayed.

Marketing

This section describes the management interfaces of the `Marketing` menu.

Featured

This section describes the featured products management interface.

Navigation / Filtering

You can filter the available products within the `Products` section:



- Use the navigation link to navigate through the displayed products.
- To filter the products by name fill in the text box.
- To filter the products by category select the category from the select box.

Add featured products

To add featured products check the check boxes beside the products you want to add (within the `Products` section) and click on `Add to featured products`.

Topseller

Topseller

Products

Sauna & Wellness

First Previous 1 / 30 Next Last

<input type="checkbox"/> Name	SKU	Active	Sub type
<input type="checkbox"/> Whirlpool / Spa Supreme II		✓	Variant
<input type="checkbox"/> Whirlpool / Spa Supreme III		✓	Variant
<input type="checkbox"/> Whirlpool / Spa Supreme III		✓	Variant
<input type="checkbox"/> Whirlpool / Spa Aura III		✓	Variant
<input type="checkbox"/> Whirlpool / Spa Supreme II		✓	Variant
<input type="checkbox"/> Whirlpool / Spa Aura I		✓	Variant

Add to topseller

Update featured products

To update featured products change the entries in question and click on the `Save featured` button (see image below).

Remove featured products

To remove featured products check the check boxes beside the products you want to remove (within the `Featured products` section) and click on `Remove from featured`.

Topseller

<input type="checkbox"/> Name	Position	SKU	Active	Sub type
<input type="checkbox"/> Infrarotkabine NORDIA SUN	<input type="text" value="1"/>	15-001-04-000-00005	✓	Standard
<input type="checkbox"/> Infrarotkabine HEARTBEAT	<input type="text" value="2"/>	15-001-04-000-00002	✓	Standard
<input type="checkbox"/> Infrarotkabine SYMPHONY	<input type="text" value="3"/>	15-001-04-000-00004	✓	Standard
<input type="checkbox"/> Infrarotkabine MOONLIGHT	<input type="text" value="4"/>	15-001-04-000-00003	✓	Standard
<input type="checkbox"/> Infrarotkabine HARMONY mit Eckelinstieg	<input type="text" value="5"/>	15-001-04-000-00001	✓	Standard
<input type="checkbox"/> Infrarotkabine SUNDOWN	<input type="text" value="6"/>	15-001-04-000-00006	✓	Standard
<input type="checkbox"/> Gerätehaus Elegant CA 015	<input type="text" value="7"/>	04-002-03-001-00002	✓	Standard
<input type="checkbox"/> Gerätehaus Elegant CA 027	<input type="text" value="8"/>	04-002-03-001-00005	✓	Standard
<input type="checkbox"/> Gerätehaus Elegant CA 021	<input type="text" value="9"/>	04-002-03-001-00004	✓	Standard

Save topseller

Remove from topseller

See also

Marketing Features

Top seller

Navigation / Filtering

You can filter the available products within the `Products` section:

- Use the navigation link to navigate through the displayed products.
- To filter the products by name fill in the text box.
- To filter the products by category select the category from the select box.

Add top sellers

To add top sellers check the check boxes beside the products you want to add (within the `Products` section) and click on `Add to topseller`.

<input type="checkbox"/> Name	SKU	Active	Sub type
<input type="checkbox"/> Whirlpool / Spa Supreme II		✓	Variant
<input type="checkbox"/> Whirlpool / Spa Supreme III		✓	Variant
<input type="checkbox"/> Whirlpool / Spa Supreme III		✓	Variant
<input type="checkbox"/> Whirlpool / Spa Aura III		✓	Variant
<input type="checkbox"/> Whirlpool / Spa Supreme II		✓	Variant
<input type="checkbox"/> Whirlpool / Spa Aura I		✓	Variant

Update top sellers

To update top sellers change the entries in question and click on the `Save topseller` button (see image below).

Remove top sellers

To remove top sellers check the check boxes beside the products you want to remove (within the `Topseller` section) and click on `Remove from topseller`.

Topseller

<input type="checkbox"/> Name	Position	SKU	Active	Sub type
<input type="checkbox"/> Infrarotkabine NORDIA SUN	<input type="text" value="1"/>	15-001-04-000-00005	✓	Standard
<input type="checkbox"/> Infrarotkabine HEARTBEAT	<input type="text" value="2"/>	15-001-04-000-00002	✓	Standard
<input type="checkbox"/> Infrarotkabine SYMPHONY	<input type="text" value="3"/>	15-001-04-000-00004	✓	Standard
<input type="checkbox"/> Infrarotkabine MOONLIGHT	<input type="text" value="4"/>	15-001-04-000-00003	✓	Standard
<input type="checkbox"/> Infrarotkabine HARMONY mit EckEinstieg	<input type="text" value="5"/>	15-001-04-000-00001	✓	Standard
<input type="checkbox"/> Infrarotkabine SUNDOWN	<input type="text" value="6"/>	15-001-04-000-00006	✓	Standard
<input type="checkbox"/> Gerätehaus Elegant CA 015	<input type="text" value="7"/>	04-002-03-001-00002	✓	Standard
<input type="checkbox"/> Gerätehaus Elegant CA 027	<input type="text" value="8"/>	04-002-03-001-00005	✓	Standard
<input type="checkbox"/> Gerätehaus Elegant CA 021	<input type="text" value="9"/>	04-002-03-001-00004	✓	Standard

See also

Marketing Features

Discounts

This section describes the management interface for discounts.

Site Actions

Add Discount Adds a new discount to the shop.

Delete Discount Deletes the currently display discount.

Data

Name The name of the discount, which is displayed as order item to the customer.

Value The value of the discount.

Type Absolute or percentaged. The type of the above entered value. If percentaged is selected the discount is based on the total price of all products within the cart.

Tax The included tax of the discount (optional)

SKU: The unique id of the discount.

Sums up: Determine if discount can be summed up with other discounts/voucher. To calculate final discount on order:

- value of discounts/voucher able to sum up is calculated
- value of highest discount/voucher that cannot be summed up is calculated
- higher of two above values is used

Criteria

Within this tab you can assign criteria to the discount. The discount is only given if a customer meets all criteria.

See Also

- *Criteria*

Vouchers

This section describes the management interface for vouchers.

Site actions

Add Voucher Group Adds a new voucher group to the shop.

Delete Voucher Group Deletes the currently displayed voucher group.

Tabs

Data

Within the data tab you can change the name of the voucher group.

Name The name of the voucher group (this is just for internal usage).

Position Voucher groups are ordered by position, lower number comes first.

Vouchers

Within the vouchers tab you can manage the vouchers of the current voucher group.

Amount The amount of voucher which are supposed to be created.

Value The value of the voucher. This is either an absolute or an percentage value dependent on the type of the voucher (see below).

Start/End: The range when the vouchers are valid.

Kind of: The type of the voucher. Either absolute or percentage.

Effective from: The minimum cart price from which the vouchers are valid. This is only the total cart item prices without shipping and payment costs.

Tax The tax of the vouchers. If you don't know the tax just let it empty.

Limit Determines how often the vouchers can be used.

Sums up Determine if Voucher can be summed up with discounts. To calculate final discount on order:

- value of discounts/voucher able to sum up is calculated
- value of highest discount/voucher that cannot be summed up is calculated
- higher of two above values is used

Add Vouchers

In order to add vouchers, fill in the provided form and click on `Add Vouchers`.

Delete Vouchers

To delete vouchers check all voucher you want to delete and click on `Delete Vouchers`.

Options

Within the options tab you can change the options for voucher numbers.

Number prefix The prefix of the voucher number: `PREFIX-xxxxx`.

Number suffix The suffix of the voucher number: `xxxxx-SUFFIX`.

Number length The length of the voucher number.

Number letters The used letters to create a voucher number

Note: The current options are only effective for upcoming vouchers.

See Also

- *Vouchers in General*

Rating Mails

Here you can send your customers invitations to rate their bought products.

All customers with an order that is `closed` for at least 14 days will get a mail with links to a rating form to all bought products of this order.

You can safely send the mails more than once, LFS will store all orders which have already got an e-mail and don't send the mail again.

In order to do that, just click on the `Send Rating Mails` button.

Options

Test The mails will get **only** to the shop notification e-mail addresses.

Send copy The mails will get to the customers and a copy of each mail is send to the notification e-mail addresses.

Utilities

This section describes the management interfaces of the `Utils` menu.

Miscellaneous

Overview

Here are some miscellaneous utilities for LFS. Actually you should never have to use theme.

Clear Cache

LFS provides an aggressive caching strategy for all content like categories and products (dependent on your environment). Actually LFS deletes out dating content automatically. However if you experience old content you can always empty the cache manually.

Set Category Levels

LFS manages internal category levels. Here you can create them safely new if something went wrong. However, you should never have to use this.

Export

Within this tab you can export selections of products.

Site Actions

Add Export Adds a new export.

Delete Export Deletes the currently displayed export.

Download Export Downloads the currently displayed export.

Tabs

Data

Name The name of the product selection. This is just for internal reasons.

Slug The unique part of the selection's URL. This is used to call the selection from outside.

Script The script which is used to create the export. By default there is only the `Generic` script. Developers can *add more scripts*.

Variants This defines how a product with variants is exported by default. This is either the default variant, the cheapest variant or all variants.

Position The position of the selection within the management interface.

Products

Within this tab you can select which products are supposed to be exported.

To export whole categories just select the check box beside the category you want to export. If you want just a sub category or single products of category, click on the category to expand the children.

For every category you can overwrite the default settings which variant(s) of a product with variants will be exported. This is either the default variant, the cheapest variant or all variants.

See Also

- *How to create a export script*

How-tos

How To Add a Horizontal Menu

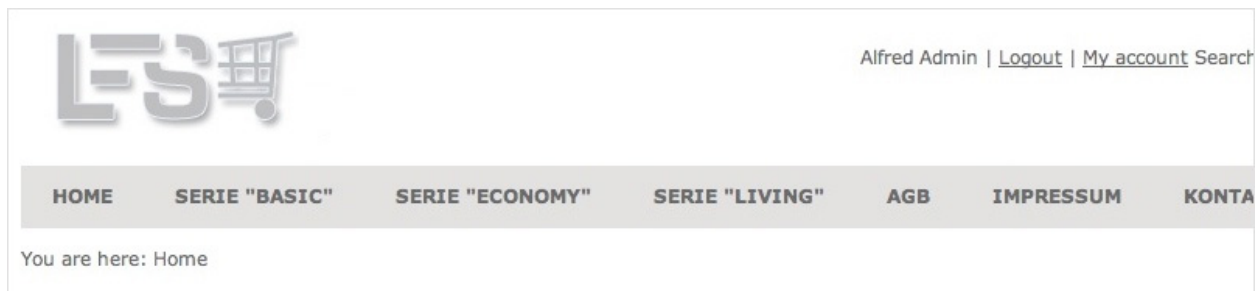
Overview

In this how-to you will learn how to add a horizontal menu.

Note: If you don't have a working LFS instance yet, you can just use our demo shop at <http://demo.getlfs.com>. Please be aware that we reset the database every two hours.

Overview

By default there is no automatically horizontal menu. You can add so-called actions to add one.



Step by Step

1. Go to Management / Shop / Actions.
2. Click on Add action.
3. Enter the Link for the action.
4. Select the Group Tabs.
5. Click on Add.
6. Now check the Active checkbox. Note: only active actions are displayed.
7. Repeat Steps 2. - 6. for every action.

Note: In order to sort action grab the handle on the left side of an action (six dots) and drag'n drop it to the new position.

How To Add a Product with Variants

Overview

In this how-to you will learn how to add a product with variants.

Note: If you don't have a working LFS instance yet, you can just use our demo shop at <http://demo.getlfs.com>. Please be aware that we reset the database every two hours.

Steps

1. Go to the LFS Management Interface.
2. Go to Catalog / Products.
3. Click on Add Product in order to add new product.
4. Enter the Name, and the Slug of the product and click on Add product.
5. Now you can enter further data as you would do for standard products for instance the price. Please note that this are in case of a Product with Variants just default values, which can be overwritten for every single variant later.
6. Change the product's type from Standard to Product with Variants and click change. You will notice that there is now a Variants tab.
7. Go to the Variants tab.
8. Click on the pencil left beside the Local Properties title in order to display the local properties add form.
9. Enter Color in the text field and click on Add Property.

Note: You can also use *global properties* to create variants, but this is beyond of these how-to.

10. Enter Red into the now provided option field and click on Add Option.
11. Repeat step 10 with Green and Blue.

Note: For convenience you can also add Red, Green, Blue to add all options at once.

12. Now go to the Variants section and select All below Color option. Click on Add Variants(s). This will create all variants based on the option you have entered above.
13. Click on the pencil of a variant in order to open its edit form.
14. By default the data is inherited from the parent. In order to override a field activate it - check the check box beside the field - and enter some information to it for instance for the price of a variant.
15. Repeat that for every variant you want to change.
16. Click on the Base Article site action in order to go back to the parent product.
17. Go to the Variants tab. You should be there automatically.
18. Check the Active check box. This will check the active check boxes of all variants and click on the Save button.

19. Select the default variant by checking one of the radio boxes below the `Default` title. This variant is displayed if the shop customer visits the `Product with Variant`.
20. Go to the `Product` tab.
21. Check the `Active` check box. This will activate the whole `Product with Variants`.

That's it

Now click on `Goto Product` and you will see your newly created `Product with variants`. There is a `Variants` section from which the customer can select the variants the product provides. All information of the product (which has been overwritten by a variant) are automatically updated if the customer choose a variant.

What's next

- Add more properties and options in order to create more complex variants.
- Use *global properties* in order to create variants.
- Check the difference between the both display types list and select.

See also

- *Products Concept*
- *Properties Concept*

How To Add a Configurable Product

Overview

In this how-to you will learn how to add a configurable product.

Note: If you don't have a working LFS instance yet, you can just use our demo shop at <http://demo.getlfs.com>. Please be aware that we reset the database every two hours.

Steps

Add Properties

A configurable product is based on *properties*. So we need to enter them at first.

1. Go to the management interface.
2. Go to the `Properties / Properties` menu.
3. Click on the `Add Property` site action and enter the name of the property, for instance "Hard drive". Then click on the `Add Property` button.
4. Now check `Configurable` within the data form of the property and click on `Save Property`.
5. Change the `Field Type` to `Select Field` and click on `Save Property Type`.
6. Now add the options, for instance:

- Name: 250GB / Price: 50.0
 - Name: 500GB / Price: 100.0
 - Name: 750GB / Price: 150.0
7. Go to Properties / Property Groups
 8. Click on the `Add Property Group` site action and enter the name of the property group, for instance “PC”. Then click the `Add Property Group` button.
 9. Go to the `Properties` tab, select the properties “Hard drive” and “RAM” and click on `Assign Properties`.

Add the Product

10. Go to the management interface.
11. Go to `Catalog / Products` menu.
12. Click on `Add product` in order to add new product.
13. Enter the name and the slug and click on `Add product`.
14. Enter further data as you would do for standard products, like prices, descriptions and so on.
15. Change the product type from `Standard` to `Configurable` and click `Change`.
16. Go to the `Properties` tab.
17. Select the `PC` property group and click on `Update Property Group`.
18. Select the default values of the properties and click on `Update Default Values`. The default values will be selected when a customer views the product.

That's it

Now click on `Goto Product` and you will see that the product has a select box from which the customer can select the options you provide above. The prices are automatically updated when the options are changed. If the customer adds the product to the cart, the selected option is stored on the product.

What's next

- Add more properties and options (see steps 3-7 on the `Add Properties` section above).
- Add a float field to the configurable product and see how the customer can be asked to enter a decimal number.
- Add a text field to the configurable product and see how the customer can be asked to enter a text.

See Also

- *Product Concepts*
- *Property Concepts*

How To Add a Product with Filters

Overview

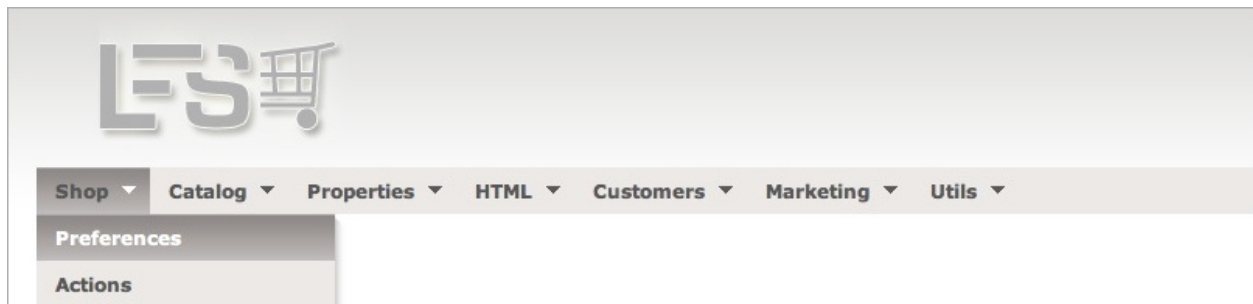
In this how-to you will learn how to create a product with filters. At the end of it you will have added a property group a property, a category with three products and a filter portlet.

Note: If you don't have a working LFS instance yet, you can just use our demo shop at <http://demo.getlfs.com>. Please be aware that we reset the database every two hours.

Steps

First we will add the corresponding filter portlet:

1. Go to Management / Shop / Preferences.



2. Go to the “Portlets” tab.
3. Select `Filter` portlet and click `Add portlet`. Enter *Filter* in the `Title` field and click `Save portlet`. You should now see the filter portlet within the `Left slot` section.

Slot: Left		
Type	Title	Modify
Filter	Filter	Delete / Edit

Now we will add properties, which are the base for the product filters.

4. Go to Management / Properties / Properties.
5. Add a new property by clicking `Add property`. (If there is no property. at all yet you will see the *Add property form* automatically).
6. Enter the Name of the property, in our case *Size* and click `Add property`.
7. Go to `Field type`, select `Select field` and click `Save property type`.
8. Now go to `Options` and fill in the `Name` field. In our case *Small*. Click `Add option`.
9. Repeat step 8 for *Middle* and *Large*. Now you should see three options within property *Size*: *Small*, *Middle* and *Large*.

Field type	
Attention: If the field type is changed all entered product values for this property will be deleted.	
Select field <input type="button" value="v"/>	<input type="button" value="Save property type"/>
Options	
Position Name <input type="text"/> <input type="button" value="Add option"/>	
1	Small <input type="button" value="X"/>
2	Middle <input type="button" value="X"/>
3	Large <input type="button" value="X"/>
<input type="button" value="Update options"/>	

Now we will add a property group.

Note: Every property has to belong exactly to one property group in order to use it with a product.

10. Got to Management / Properties / Property Groups.
11. Add a new property group by clicking the Add property group button. (If there is no property group at all yet you will see the *Add property group* form automatically).
12. Enter the Name of the property, in our case *T-Shirts* and click Add property group.

Now assign the property to the property group:

13. Go to the “Properties” tab.
14. Select *Size* and click Assign properties.

You should now see *Size* within Assigned properties.

Assigned properties	
<input type="checkbox"/>	Position Name
<input type="checkbox"/> 1	Size
<input type="button" value="Remove properties"/> <input type="button" value="Update properties"/>	

We will now add the Property group to some products.

15. Go to Management / Catalog / Products.
16. Add a new product clicking Add product. (If there is no product at all yet you will see the *Add product* form automatically).
17. Fill in Name *T-Shirt One* and Price *10.00* and click Add product.

Now we assign values to the properties:

18. Go to the Properties tab of that product.
19. Select our Property group *T-Shirts* and click Update property groups You will now see the Property *Size*.
20. Select *Size Small*.

21. Repeat Steps 16 - 20 for *T-Shirt Two* and *T-Shirt Three* and assign sizes *Middle* and *Large* and prices *100.00* and *1000.00*. Now you should have three products *T-Shirt One*, *T-Shirt Two* and *T-Shirt Three* with sizes *Small*, *Middle* and *Large* and prices *10.00*, *100.00* and *1000.00*.

We need to make the products active:

22. Go to Management / Catalog / Products Overview.
23. Select *Active* for all products and click *Save*.

URL	Name	SKU	Active	Sub type
<input type="checkbox"/> t-shirt-one	T-Shirt One		<input type="checkbox"/>	Standard
<input type="checkbox"/> t-shirt-two	T-Shirt Two		<input type="checkbox"/>	Standard
<input type="checkbox"/> t-shirt-three	T-Shirt Three		<input type="checkbox"/>	Standard

Now we add a new category *T-Shirts*:

Note: Filtering takes place on products of a category, hence we add a category for our newly products.

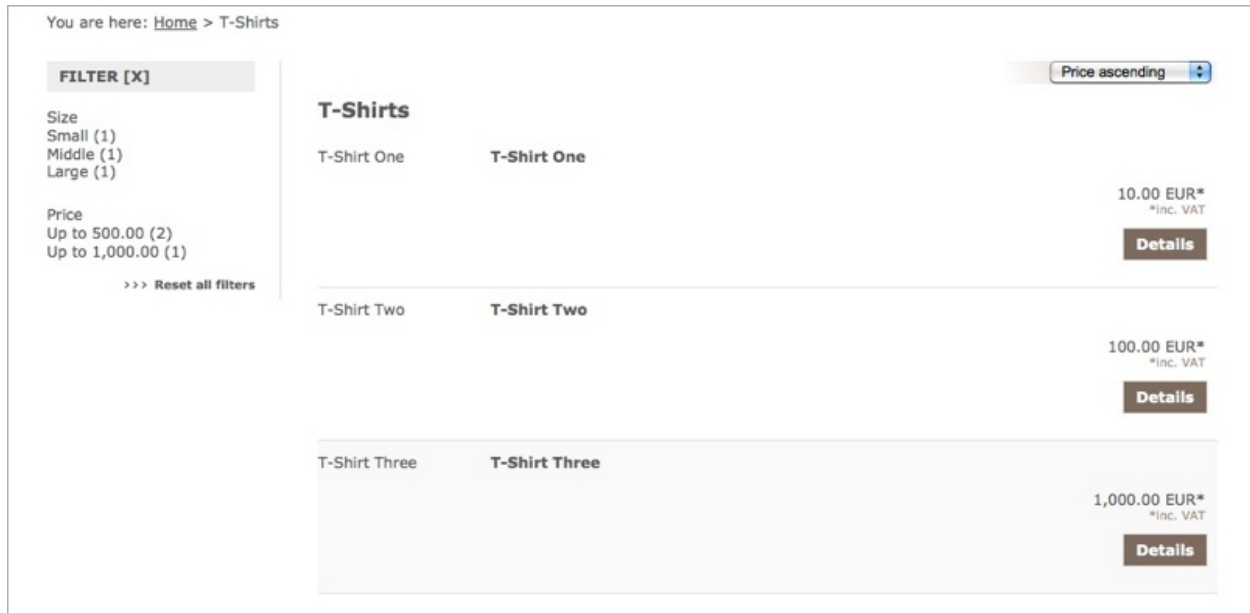
24. Go to Management / Catalog / Categories.
25. Add a new category by clicking “Add category”. (If there is no category at all yet you will see the “Add category” form automatically).
26. Fill in the Name: *T-Shirts* and click *Add category*.

Now we assign the products to the category:

27. Go to the products tab of that category.
28. Select the products *T-Shirt One*, *T-Shirt Two*, *T-Shirt Three* and click *Add to category*.

Now we are ready to preview our new content:

29. Click on the *Preview* button. You should now see the filter portlet with the ability to select the sizes *Small*, *Middle* and *Large* and prices *0 - 500.00* and *501.00 - 1000.00*.



What's next?

- Add some more properties to the property group *T-Shirts*.
- Add the property group *T-Shirts* to more products.
- Assign products and values via *Property Groups*:
 1. Go to Properties / Properties Groups
 2. Select *T-Shirts*
 3. Go to *Products* tab
 4. Go to *Values* tab
- Check out the other options of the properties, e.g. the `Field` type or the `Unit` field.
- Check out creating variants on base of properties and see how filters work with variants too.

How To Manage Payment Methods

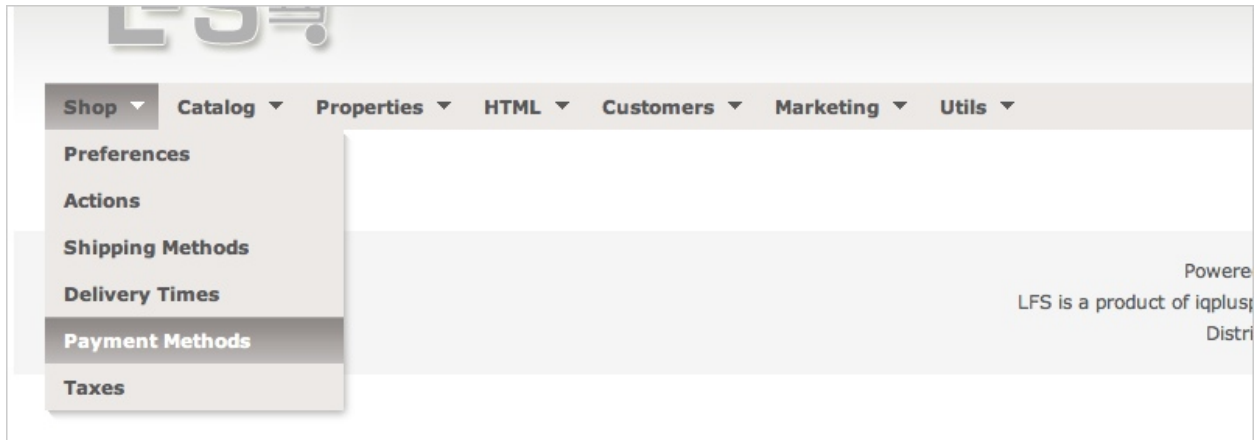
Overview

In this how-to you will learn how to add and edit payment methods and how to add criteria and prices for them.

Note: If you don't have a working LFS instance yet, you can just use our demo shop at <http://demo.getlfs.com>. Please be aware that we reset the database every two hours.

Get started

In order to add/edit a payment method please go to Management / Shop / Payment Methods.



If there are no payment methods yet, you will automatically get to the add payment method form. Otherwise the first payment method is displayed and you can click on a payment method to edit this or on `Add payment method` to add a new one.

If you have done that you can edit or enter the data for a payment method as described below.

Data

The data tab contains all core data for the payment method.

 A screenshot of the 'Vorkasse / Überweisung' (Advance payment / Bank transfer) payment method form. The form is titled 'Vorkasse / Überweisung' and has a green '+ Add payment method' button. On the left, there is a sidebar with a list of payment methods: 'Nachnahme', 'Lastschriftverfahren', 'Kreditkarte', 'PayPal', 'Per Rechnung', and 'Bar bei Abholung'. The main form area has three tabs: 'Data', 'Criteria', and 'Prices'. The 'Data' tab is selected, showing 'Core data'. It includes a 'Name:' field with the value 'Vorkasse / Überweisung', an 'Active:' checkbox which is checked, and a 'Description:' field with the text 'Zahlen Sie bequem per Banküberweisung.' Below the description field is a rich text editor toolbar with various formatting options like bold, italic, underline, list, and link.

Now fill in the fields:

- **Name:** The name of the payment method, which is displayed to the shop customer.
- **Description:** A short description of the payment method, which is also displayed to the customer.
- **Note:** A note of the payment method, which is displayed on the confirmation mail after the shop customer has been checked out.
- **Priority:** The first valid payment method with the highest priority (smaller number) method is displayed to the customer.
- **Image:** An image for the payment method, which is displayed to the shop customer.
- **Tax:** The included tax of the payment method's price.
- **Price:** The default price of the payment method. This can be overwritten within the price tab (see below).

- **Module:** The dotted name of the external package which processes the payment (this is for developers only).
- **Type:** The type of the payment method. Dependent on that additional fields for input (within the checkout process) will be displayed. There are three types at the moment:
 - **Plain** No additional fields are displayed.
 - **Bank** Fields for a bank account are displayed.
 - **Credit Card** Fields for a credit card are displayed.

And click on the `Save`-button.

Criteria

Optional you can add some criteria to the payment method. Only when all criteria are true the payment method is displayed to the shop customer.

To add criterion proceed as following:

- Click on the `Add criteria`-button (adds a criterion on first position) or on the `plus` button beside a criterion (adds a criterion below)
- Select the criteria type you want, e.g. `Weight` (this is the weight of all cart items).
- Select the operator, e.g. `Less than equal to`.
- Enter the value, e.g. `50`.
- Altogether this means the payment method is valid if the weight of all cart items is less than or equal to 50 units.
- You can add as many criteria you want.
- Click on `Save criteria`.

To update criteria proceed as following:

- Change the values of the criteria to your needs.
- Click on `Save criteria`

To delete a criterion proceed as following:

- Click on the `minus` button beside the criterion.
- Click on `Save criteria`.

Prices

Optional you can add additional prices to the payment method and restrict them with criteria. The first price which meets all criteria will be taken.

The screenshot shows the 'Prices' management interface. At the top, there are three tabs: 'Data', 'Criteria', and 'Prices'. The 'Prices' tab is selected. Below the tabs, there is a section titled 'Prices' with a text input field and an 'Add price' button. Below this is a section titled 'Existing prices' containing a table with the following data:

<input type="checkbox"/>	Priority	Price	
<input type="checkbox"/>	1	6.9	Edit criteria Country Is Deutschland Shipping Is Paketversand, Paketversand - 24h Service
<input type="checkbox"/>	2	12.9	Edit criteria Country Is Österreich Shipping Is not valid Stückgutversand - Spedition, Stückgutversand - 24h Service - Spedition, Speditionsversand - Direkt
<input type="checkbox"/>	3	17.9	Edit criteria Shipping Is Stückgutversand - Spedition, Stückgutversand - 24h Service - Spedition
<input type="checkbox"/>	4	29.9	Edit criteria Shipping Is valid Speditionsversand - Direkt

At the bottom of the 'Existing prices' section, there are two buttons: 'Delete prices' and 'Update prices'.

To add a price proceed as following:

- To manage prices go to the `Prices` tab.
- To add a new price enter the value into the text field and click `Add price`.
- To add/edit criteria for that price click on `Edit criteria` link. A pop-up window will open.
- Click on `Add criteria` and change the criteria type, the operator and the value to your needs.

To update/delete a price proceed as following:

- To update the prices change the priority and/or the value of the price and click on `Update prices`.
- To delete the prices select the check boxes of the prices you want delete and click on `Delete prices`.

See also

- [Manage payment methods](#)

How To Manage Shipping Methods

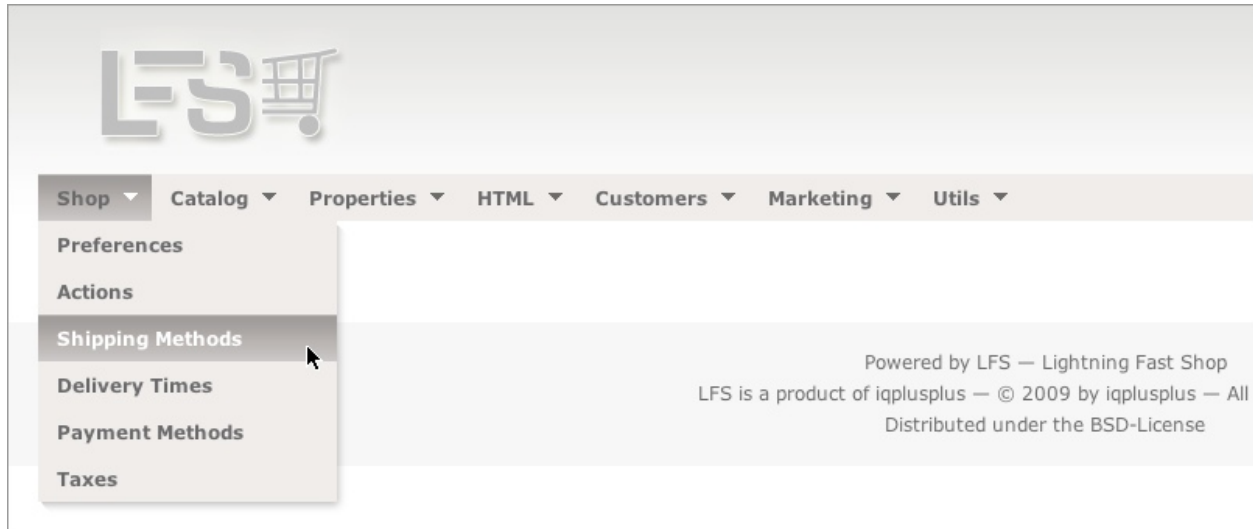
Overview

In this how-to you will learn how to add and edit shipping methods and how to add criteria and prices for them.

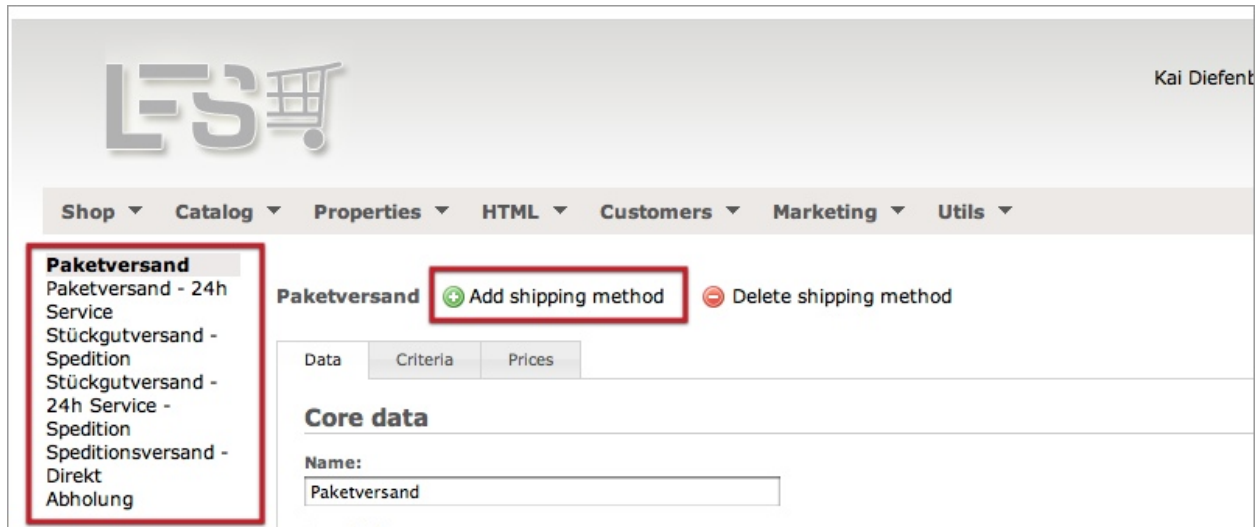
Note: If you don't have a working LFS instance yet, you can just use our demo shop at <http://demo.getlfs.com>. Please be aware that we reset the database every two hours.

Get Started

In order to add/edit a shipping method please go to Management / Shop / Shipping Methods.



If there are no shipping methods yet, you will automatically get to the add shipping method form. Otherwise the first shipping method is displayed and you can click on a shipping method to edit this or on Add shipping method to add a new one.



If you have done that you can edit or enter the data for a shipping method as described below.

Data

The data tab contains all core data for the shipping method.

Paketversand

Paketversand - 24h Service
Stückgutversand - Spedition
Stückgutversand - 24h Service -
Spedition
Speditionsversand - Direkt
Abholung

Paketversand
+ Add shipping method
- Delete shipping method

Data

Criteria

Prices

Core data

Name:

Description:

B I U |
 [List Icons] |
 [Table Icon] |
 [Image Icon] |
 [Undo] [Redo] |
 HTML [Link Icon] [Unlink Icon] |
 Styles ▼

Die Lieferung erfolgt per Paketdienst.

- Now fill in the fields:
 - **Name:** The name of the shipping method, which is displayed to the shop customer.
 - **Description:** A short description of the shipping method, which is also displayed to the customer.
 - **Note:** A note of the shipping method, which is displayed on the confirmation mail after the shop customer has been checked out.
 - **Priority:** The first valid shipping method with the highest priority (smaller number) method is displayed to the customer.
 - **Image:** An image for the shipping method, which is displayed to the shop customer.
 - **Tax:** The included tax of the shipping method's price.
 - **Price:** The default price of the shipping method. This can be overwritten within the price tab (see below).
- Click on **Save**-button

Criteria

Optional you can add some criteria to the shipping method. Only when all criteria are true the shipping method is displayed to the shop customer.

Paketversand

Paketversand - 24h Service
Stückgutversand - Spedition
Stückgutversand - 24h Service - Spedition
Speditonsversand - Direkt
Abholung

Paketversand

Add shipping method

Delete shipping method

Data

Criteria

Prices

Criteria

Add criterion

10

Width

Less than equal to

60.0

20

Height

Less than equal to

80.0

30

Length

Less than equal to

200.0

40

Combined length and girth

Less than equal to

300.0

50

Weight

Less than equal to

50.0

Save criteria

To add criterion proceed as following:

- Click on the `Add criteria`-button (adds a criterion on first position) or on the `plus` button beside a criterion (adds a criterion below)
- Select the criteria type you want, e.g. `Weight` (this is the weight of all cart items).
- Select the operator, e.g. `Less than equal to`.
- Enter the value, e.g. `50`.
- Altogether this means the shipping method is valid if the weight of all cart items is less than or equal to 50 units.
- You can add as many criteria you want.
- Click on `Save criteria`.

To update criteria proceed as following:

- Change the values of the criteria to your needs.
- Click on `Save criteria`

To delete a criterion proceed as following:

- Click on the `minus` button beside the criterion.
- Click on `Save criteria`.

Prices

Optional you can add additional prices to the shipping method and restrict them with criteria. The first price which meets all criteria will be taken.

The screenshot shows the configuration interface for the shipping method 'Paketversand'. On the left, a list of shipping methods is shown, with 'Paketversand' selected. The main area has three tabs: 'Data', 'Criteria', and 'Prices'. The 'Prices' tab is active, showing a section titled 'Prices' with a text input field and an 'Add price' button. Below this is a section titled 'Existing prices' containing a table with columns 'Priority' and 'Price'. One entry is shown with a priority of 3 and a price of 19.9. To the right of the price field is an 'Edit criteria' link. Below the table, the criteria are listed: 'Country Is GERMANY, SWITZERLAND' and 'Weight: Less than or equal to 40.0'. At the bottom of the 'Existing prices' section are 'Delete prices' and 'Update prices' buttons.

To add a price proceed as following:

- To manage prices go to the `Prices` tab.
- To add a new price enter the value into the text field and click `Add price`.
- To add/edit criteria for that price click on `Edit criteria` link. A pop-up window will open.
- Click on `Add criteria` and change the criteria type, the operator and the value to your needs.

To update/delete a price proceed as following:

- to update the prices change the priority and/or the value of the price and click on `Update prices`.
- To delete the prices select the check boxes of the prices you want delete and click on `Delete prices`.

See Also

- *Manage shipping methods*

How To Setup Paypal

Overview

In this how-to you will learn how to setup paypal for your shop

Steps

1. Setup a paypal developer account at <https://developer.paypal.com/>
2. Add a pre-configured test Buyer https://developer.paypal.com/cgi-bin/devscr?cmd=_sandbox-acct-session
3. Add a pre-configured test Seller https://developer.paypal.com/cgi-bin/devscr?cmd=_sandbox-acct-session
4. Launch the sandbox account for the Seller from https://developer.paypal.com/cgi-bin/devscr?cmd=_sandbox-acct-session
5. Set up IPN
 - Click on “Profile” - “Instant Payment Notification Preferences”
 - Set the Notification URL to <http://www.yourdomainname.com/paypal/ipn/>
 - Turn IPN On
6. Set up PDT
 - Click on “Profile” - “Website Payment Preferences”
 - Turn on “Auto Return”
 - Set the “Return URL” to <http://www.yourdomainname.com/paypal/pdt/>
 - Set “Payment Data Transfer” to On, this will create an Identity Token for us.
7. Copy our seller information to settings.py e.g.:

```
PAYPAL_RECEIVER_EMAIL = "seller_1262786866_biz@yourdomainname.com"
PAYPAL_IDENTITY_TOKEN = "j0Iw3M4l6znE45kWkyQs43PkwC9bkaceteiWXfddg5q_
↪CW1Ev4HGuqVPPfBG"
```

8. Deploy your site to a live internet site for testing (Paypal servers must be able to see your site).
9. When you are finished testing your site and ready to go live, set up a live Paypal Business account (Website Payments Standard account has been reported to work) and repeat steps 5-8

How To Add a Export Script

Overview

LFS provides a generic export engine for products (*see here for more*). In this tutorial you will learn how to create your own scripts to format the data like you want to.

Note: If you don't have a working LFS instance yet, you can just use our demo shop at <http://demo.getlfs.com>. Please be aware that we reset the database every two hours.

Steps

In order to create a new export script you should first create a new Django application (or use an existing one). This is beyond this tutorial. If you do not know how to do this, please refer to the excellent [Django tutorial](#).

Within the `__init__.py` of your application create a function that will return the products, like so:

```

1  # python imports
2  import csv
3
4  # django imports
5  from django.http import HttpResponse
6  from django.core.mail import send_mail
7
8  # lfs imports
9  from lfs.export.utils import register
10
11 def export(request, export):
12     """Export method for acme.com
13     """
14     response = HttpResponse(mimetype="text/csv")
15     response["Content-Disposition"] = "attachment; filename=acme.txt"
16
17     writer = csv.writer(response, delimiter=";", quotechar='"', quoting=csv.QUOTE_ALL)
18
19     for product in export.get_products():
20         writer.writerow(product.id, product.get_name())
21
22     return response
23
24 register(export, "acme.com")

```

The code explained

1-6 Some simple Python and Django imports. These may vary for your code.

9 Imports the register method to register your script.

11 The function which will contain your code to create the exported data.

14/15 We decided to give a response with a download back. Your code may vary here.

16 We decided to use Python csv modules. Your code may vary here.

19 All selected products can be get with the `get_products` method of the passed export object.

24 The registration of your function. This line must be called while Django is starting up.

Getting ready

Now you can go the management interface, create a new export, select the products and your newly script and call it via the `Export` button.

You might want to create a cron job which calls your script regularly. (*See [here for more](#)*)

How To Use Celery

Overview

Celery is a distributed task queue that can be used to send e-mails from LFS asynchronously.

There is no special integration in LFS for Celery but there are general patterns for Django based projects that can be used to get asynchronous e-mail backend.

Dependencies

You need:

- `celery`
- `django-celery-email`

as well as some Celery backend, eg Redis. Consult Celery documentation for details.

Installation

Follow the Celery's [first steps with Django](#) and `django-celery-email` [documentation](#).

Developing

Developing LFS

This section describes how to develop LFS.

Warning: This is only for development. Don't use this version in production. We might break the head, add stuff which need database migrations or introduce some security issues, etc.

Creating a Development Environment

There is an installer based on [zc.buildout](https://github.com/diefenbach/lfs-buildout-development.git), which should make the installation straightforward:

1. `$ git clone https://github.com/diefenbach/lfs-buildout-development.git`
2. `$ cd lfs-buildout-development`
3. `$ python bootstrap.py`
4. `$ bin/buildout -v`
5. `$ bin/django syncdb`
6. `$ bin/django migrate`
7. `$ bin/django lfs_init`
8. `$ bin/django test lfs.core`
9. `$ bin/django runserver`
10. Browse to <http://localhost:8000>

Note: You might want to fork LFS on [GitHub](#) and point to it within `buildout.cfg` first.

Contributing Code to the Core

If you consider to contribute code to LFS, please read the following statements first:

1. First of all, you are very welcome!
2. Generally, it would be great if you would discuss new stuff first. We are very reluctant to add new things. Every new feature should have a real live use case. Find us on IRC or the [LFS Google Group](#).
3. Fork LFS [GitHub](#) and send us pull requests.
4. Please make sure that you just add related code to your fork. This makes it easier to review and pull your code.
5. The code must be put under a [permissive free software licenses](#) like BSD, otherwise we can't add it. For instance, code under the GPL or a other [copyleft software licenses](#) won't be added to the core.
6. Python code must follow [PEP 8](#). The maximum of 79 characters per line is the only exception. You may want to check your code with [pep8](#). The following statement should run without complaints:

```
$ pep8 --repeat --ignore=E501 /path/to/lfs
```

7. Every new feature must have unit tests and documentation.
8. *All tests must pass*. Please check this with:

```
$ bin/django test lfs.core
```

9. New features shouldn't make LFS slower. Please see [Benchmarking LFS](#).
10. Add yourself to [CREDITS.txt](#).

Contributing Translations

Please refer to [Contributing translations](#).

Testing LFS

This section describes how to test LFS.

Overview

Testing is based on [nose](#) and [django-nose](#).

Examples

Test everything:

```
$ bin/django test lfs.core
```

Test everything with coverage (you'll find the coverage report in the current directory):

```
$ bin/django test lfs.core --with-coverage --cover-package=lfs --cover-html
```

Test only the catalog:

```
$ bin/django test lfs.catalog
```

Test only the catalog with coverage (you'll find the coverage report in the current directory):

```
$ bin/django test lfs.catalog --with-coverage --cover-package=lfs.catalog --cover-html
```

Test only one method without capturing the output (this is helpful if you want to debug a test with pdb):

```
$ bin/django test lfs.catalog.tests:ViewsTestCase.test_file -s
```

See also

- [nose](#)
- [nose options](#)
- [django-nose](#)

Benchmarking LFS

This section describes how to benchmark LFS.

Overview

The development buildout comes with `lfs_bench`, a small application, which provides some tools to benchmark LFS.

You can use it in case you want to optimize LFS in terms of speed. In case you want to development a new feature, please compare the versions with and without this feature and try to not slow down LFS.

JMeter

1. Install [JMeter](#)
2. Prepare the database:

```
$ bin/django lfs_init  
$ bin/django lfs_generate_content_for_benchmark
```

3. Start LFS

- Start LFS in production mode, for instance with `bin/django-gunicorn`. and a reverse proxy in front of it. In other words don't use the development server.
- Set `DEBUG` to `False`

4. Start JMeter

Make sure you start it from the buildout base directory (that's the same directory you start Django from):

```
$ /path/to/jmeter/bin/jmeter -t src/lfs_bench/lfs_bench/jmeter/lfs.jmx -p ↵  
↵src/lfs_bench/lfs_bench/jmeter/user.properties
```

5. Execute the Testplan and check the result within Summary Report for instance.

Apache Benchmark - ab

1. Prepare the database:

```
$ bin/django lfs_init  
$ bin/django lfs_generate_content_for_benchmark
```

2. Start LFS

- Start LFS in production mode, for instance with `bin/django-gunicorn.` and a reverse proxy in front of it. In other words don't use the development server.
- Set `DEBUG` to `False`

3. Use `ab`, for instance like that:

```
$ ab -n 1000 -c 20 http://localhost/product-1-1-1
```

4. Check the results.

Python Profiling

The development buildout provides a middleware by default, which let you profile single requests. Just suffix the request with `?prof` in order to get profiling data, for instance:

```
http://localhost/product-1-1-1?prof
```

See also:

- [JMeter](#)
- [Apache Benchmark - ab](#)
- [Python stats reference](#)

Changing models

This section describes how to modify and create new models in LFS

Overview

Since version 0.8 LFS uses [South](#) migrations so you're encouraged to read South documentation first. If you need to modify models layer of LFS then you should use 'shemamigration' command. If you have to migrate data then you should use 'datamigration' command. See examples below.

Note: Do not forget to commit migrations into repository! They're created in `migrations/` directory of the application they're created for

Examples

New field added to the model, or modified existing field in lfs.customer:

```
$ bin/django schemamigration lfs.customer --auto
```

New model added to lfs.catalog:

```
$ bin/django schemamigration lfs.customer --auto
```

Execute migration that was just created with 'schemamigration':

```
$ bin/django schemamigration lfs.customer --auto
$ bin/django migrate lfs.customer
```

Correct migration instead of creating new one, eg. if you've created a migration and then realized that your model still has to be corrected (see more [here](#)):

```
$bin/django schemamigration lfs.customer --auto --update
```

Contributing translations

There are two preferred ways to contribute translations:

Via DVCS

If your are a (Django-)developer you might want to fork django-lfs and lfs-theme on [GitHub](#) add the translations as [used to](#), commit them and send us a pull request.

Note: Please make sure to fork the correct version branch.

Via Transifex

If your are a translator you can just go to [transifex](#), choose or add the language of your choice and start to translate via the web interface. Once you are ready we would add your translations to the source code.

Note: Please make sure to use the correct version.

Other

If none of these fit your workflow you can of course just send us the translation files.

See also

- [LFS on Transifex](#)
- [How to create language files in Django](#)

Miscellaneous

API

Plug-ins

Criterion

class `lfs.criteria.models.Criterion`

Base class for all criteria.

Attributes:

cart The current cart of the current customer.

content The content object the criterion belongs to.

operator The current selected operator for the criterion.

position The position of the criterion within a list of criteria of the content object.

product The product, if the criterion is called from a product detail view. Otherwise this is None.

request The current request.

Constants:

EQUAL, LESS_THAN, LESS_THAN_EQUAL, GREATER_THAN, GREATER_THAN_EQUAL, IS_SELECTED, IS_NOT_SELECTED Integers which represents certain operators.

INPUT, SELECT, MULTIPLE_SELECT Constants which represents the types of selectable values. One of these must be returned from `get_value_type`.

NUMBER_OPERATORS A list of operators which can be returned from `get_operators`.

```
[
    [EQUAL, _(u"Equal to")],
    [LESS_THAN, _(u"Less than")],
    [LESS_THAN_EQUAL, _(u"Less than equal to")],
    [GREATER_THAN, _(u"Greater than")],
    [GREATER_THAN_EQUAL, _(u"Greater than equal to")],
]
```

SELECTION_OPERATORS A list of operators which can be returned from `get_operators`.

```
[
    [IS_SELECTED, _(u"Is selected")],
    [IS_NOT_SELECTED, _(u"Is not selected")],
]
```

VALID_OPERATORS A list of operators which can be returned from `get_operators`.

```
[
    [IS_VALID, _(u"Is valid")],
    [IS_NOT_VALID, _(u"Is not valid")],
]
```

STRING_OPERATORS A list of operators which can be return from `get_operators`.

```
[
    [EQUAL, _(u"Equal to")],
    [CONTAINS, _(u"Contains")],
]
```

get_operators()

Returns the selectable operators of the criterion which are displayed to the shop manager. This is a list of list, whereas the first value is integer, which is stored within the criterion and the second value is the string which is displayed to the shop manager, e.g.:

```
[
    [0, _(u"Equal to")],
    [1, _(u"Less than")],
    [2, _(u"Less than equal to")],
    [3, _(u"Greater than")],
    [4, _(u"Greater than equal to")],
]
```

Note: You can use one of the provided class attributes, see above.

- NUMBER_OPERATORS
 - SELECTION_OPERATORS
 - VALID_OPERATORS
 - STRING_OPERATORS
-

get_selectable_values(request)

Returns the selectable values as a list of dictionary, see below. This is only called when `get_value_type` returns `SELECT` or `MULTIPLE_SELECT`.

```
[
    {
        "id": 0,
        "name": "Name 0",
        "selected": False,
    },
    {
        "id": 1,
        "name": "Name 1",
        "selected": True,
    },
]
```

get_template(request)

Returns the template to render the criterion.

get_value_type()

Returns the type of the selectable values field. Must return one of:

- self.INPUT
- self.SELECT
- self.MULTIPLE_SELECT

get_value()

Returns the current value of the criterion.

is_valid (*request*, *product=None*)

Returns True if the criterion is valid otherwise False.

render (*request*, *position*)

Renders the criterion as html in order to displayed it within the management form.

update (*value*)

Updates the value of the criterion.

Parameters:

value The value the shop user has entered for the criterion.

OrderNumberGenerator

class `lfs.plugins.OrderNumberGenerator`

Base class from which all order number generators should inherit.

Attributes:

`cart`

The current cart of the customer.

`customer`

The customer of the order.

`order`

The order for which a new number is generated.

`request`

The current request

`user`

The user of the order.

get_form (***kwargs*)

Returns the form which is used within the shop preferences management interface.

All parameters are passed to the form.

get_next (*formatted=True*)

Returns the next order number as string. Derived classes must implement this method.

Parameters:

formatted If True the number will be returned within the stored format, which is based on Python default string formatting operators, e.g. `%04d`.

exclude_form_fields ()

Returns a list of fields, which are excluded from the model form, see also `get_form`.

init (*request*, *order*)

Initializes the order number generator. This method is called automatically from LFS.

PaymentMethodProcessor

class `lfs.plugins.PaymentMethodProcessor` (*request, cart=None, order=None*)

Base class from which all 3rd-party payment method processors should inherit.

Attributes:

cart The current cart. This is only set, when create order time is ACCEPTED.

order The current order. This is only set, when create order time is IMMEDIATELY.

request The current request.

get_create_order_time ()

Returns the time when the order should be created. It is one of:

PM_ORDER_IMMEDIATELY The order is created immediately before the payment is processed.

PM_ORDER_ACCEPTED The order is created when the payment has been processed and accepted.

get_pay_link ()

Returns a link to the payment service to pay the current order, which is displayed on the thank-you page and the order confirmation mail. In this way the customer can pay the order again if something has gone wrong.

process ()

Implements the processing of the payment method. Returns a dictionary with several status codes, see below.

Return Values:

This values are returned within a dictionary.

accepted (mandatory) Indicates whether the payment is accepted or not. if this is `False` the customer keeps on the checkout page and gets `message` (if given) below. If this is `True` the customer will be redirected to `next_url` (if given).

message (optional) This message is displayed on the checkout page, when the order is not accepted.

message_location (optional) The location, where the message is displayed.

next_url (optional) The url to which the user is redirect after the payment has been processed. if this is not given the customer is redirected to the default thank-you page.

order_state (optional) The state in which the order should be set. It's just PAID. If it's not given the state keeps in SUBMITTED.

PriceCalculator

class `lfs.plugins.PriceCalculator` (*request, product, **kwargs*)

This is the base class that pricing calculators must inherit from.

Attributes:

product The product for which the price is calculated.

request The current request.

get_base_price (*with_properties=True*)

Returns the base price of the product.

Parameters:

with_properties If the instance is a configurable product and with_properties is True the prices of the default properties are added to the price.

get_base_price_net (*with_properties=True*)

Returns the net base price of the product.

Parameters:

with_properties If the instance is a configurable product and with_properties is True the prices of the default properties are added to the price.

get_base_price_gross (*with_properties=True*)

Returns the gross base price of the product.

Parameters:

with_properties If the instance is a configurable product and with_properties is True the prices of the default properties are added to the price.

get_base_packing_price (*with_properties=True*)

Returns the base packing price of the product.

Parameters:

with_properties If the instance is a configurable product and with_properties is True the prices of the default properties are added to the price.

get_base_packing_price_net (*with_properties=True*)

Returns the base packing net price of the product.

Parameters:

with_properties If the instance is a configurable product and with_properties is True the prices of the default properties are added to the price.

get_base_packing_price_gross (*with_properties=True*)

Returns the base packing gross price of the product.

Parameters:

with_properties If the instance is a configurable product and with_properties is True the prices of the default properties are added to the price.

get_customer_tax (*with_properties=True*)

Returns the calculated tax for the current customer and product.

Parameters:

with_properties If the instance is a configurable product and with_properties is True the taxes of the default properties are added to the price.

get_customer_tax_rate ()

Returns the tax rate for the current customer and product.

get_effective_price ()

Effective price is used for sorting and filtering. Usually it is same as value from get_price but in some cases it might differ (eg. if we add eco tax to product price)

get_for_sale_price (*with_properties=True*)

Returns the sale price for the product.

Parameters:

with_properties If the instance is a configurable product and with_properties is True the prices of the default properties are added to the price.

get_for_sale_price_net (*with_properties=True*)

Returns the sale net price for the product.

Parameters:

with_properties If the instance is a configurable product and `with_properties` is `True` the prices of the default properties are added to the price.

get_for_sale_price_gross (*with_properties=True*)

Returns the sale net price for the product.

Parameters:

with_properties If the instance is a configurable product and `with_properties` is `True` the prices of the default properties are added to the price.

get_price (*with_properties=True*)

Returns the stored price of the product without any tax calculations. It takes variants, properties and sale prices into account, though.

Parameters:

with_properties If the instance is a configurable product and `with_properties` is `True` the prices of the default properties are added to the price.

get_price_net (*with_properties=True*)

Returns the net price of the product.

Parameters:

with_properties If the instance is a configurable product and `with_properties` is `True` the prices of the default properties are added to the price.

get_price_gross (*with_properties=True*)

Returns the real gross price of the product. This is the base of all price and tax calculations.

Parameters:

with_properties If the instance is a configurable product and `with_properties` is `True` the prices of the default properties are added to the price.

get_standard_price (*with_properties=True*)

Returns always the stored standard price for the product. Independent whether the product is for sale or not. If you want the real price of the product use `get_price` instead.

Parameters:

with_properties If the instance is a configurable product and `with_properties` is `True` the prices of the default properties are added to the price.

get_standard_price_net (*with_properties=True*)

Returns always the standard net price for the product. Independent whether the product is for sale or not. If you want the real net price of the product use `get_price_net` instead.

Parameters:

with_properties If the instance is a configurable product and `with_properties` is `True` the prices of the default properties are added to the price.

get_standard_price_gross (*with_properties=True*)

Returns always the gross standard price for the product. Independent whether the product is for sale or not. If you want the real gross price of the product use `get_price_gross` instead.

Parameters:

with_properties If the instance is a configurable product and `with_properties` is `True` the prices of the default properties are added to the price.

get_product_tax (*with_properties=True*)

Returns the calculated tax for the current product independent of the customer.

get_product_tax_rate ()

Returns the stored tax rate of the product. If the product is a variant it returns the parent's tax rate.

price_includes_tax ()

Returns `True` if stored price includes tax. `False` if not.

ShippingMethodPriceCalculator

class `lfs.plugins.ShippingMethodPriceCalculator` (*request, shipping_method*)

Base class from which all 3rd-party shipping method prices should inherit.

Attributes:

request The current request.

shipping_method The shipping method for which the price is calculated.

get_price ()

Returns the stored price without any calculations.

get_price_gross ()

Returns the gross price of the shipping method.

get_price_net ()

Returns the net price of the shipping method.

get_tax ()

Returns the total tax of the shipping method.

get_tax_rate ()

Returns the tax rate of the shipping method.

Settings

These are settings specific for LFS which can be changed within `lfs_project/settings`. For Django's default settings which are also relevant for LFS, please visit [Django settings](#) for an explanation.

Plug-ins

LFS_ORDER_NUMBER_GENERATOR The class which is responsible for the creation of order numbers. LFS ships with: `lfs_order_numbers.models.OrderNumberGenerator`.

See also:

[How to add an own order numbers generator](#)

LFS_PAYMENT_METHOD_PROCESSORS List of list of available 3rd-party payment method processors, whereas the first entry is the dotted name to a `PaymentMethod` and the second entry is the name, which is displayed. These are provided for selection within the payment method management interface, e.g.:

```
LFS_PAYMENT_METHOD_PROCESSORS = [
    ["acme.ACMEPaymentMethod", _("ACME payment")],
]
```

See also:

How to add an own payment processor

LFS_PRICE_CALCULATORS List of list of available price calculators, whereas the first entry is the dotted name to a PriceCalculator and the second entry is the name, which is displayed. These are provided for selection within the shop preferences and the product. LFS is shipped with following entries:

```
LFS_PRICE_CALCULATORS = [
    ["lfs.gross_price.calculator.GrossPriceCalculator", _(u"Price includes tax")],
    ["lfs.net_price.calculator.NetPriceCalculator", _(u"Price excludes tax")],
]
```

See also:

How to add custom product pricing module

LFS_SHIPPING_METHOD_PRICE_CALCULATORS List of list of available shipping method price calculators, whereas the first entry is the dotted name to a ShippingMethodPriceCalculator and the second entry is the name, which is displayed. These are provided for selection within the shipping method. LFS is shipped with following entries:

```
LFS_SHIPPING_METHOD_PRICE_CALCULATORS = [
    ["lfs.shipping.GrossShippingMethodPriceCalculator", _(u'Price includes tax')],
    ["lfs.shipping.NetShippingMethodPriceCalculator", _(u'Price excludes tax')],
]
```

See also:

How to add an own shipping price calculator

Miscellaneous

LFS_AFTER_ADD_TO_CART URL name to which LFS redirects after the customer has put a product into the cart. LFS ships with `lfs_added_to_cart`, which displays the last product, which has been added to the cart. A reasonable alternative is `lfs_checkout_dispatcher`, which redirects directly to the checkout view.

LFS_CRITERIA List of list of available criteria, whereas the first entry is the dotted name to a criterion and the second entry is the name of the criterion, which is displayed to the users. These criteria are provided to a shop manager for selection for on several locations. LFS is shipped with following criteria:

```
LFS_CRITERIA = [
    ["lfs.criteria.models.CartPriceCriterion", _(u"Cart Price")],
    ["lfs.criteria.models.CombinedLengthAndGirthCriterion", _(u"Combined Length_
↪and Girth")],
    ["lfs.criteria.models.CountryCriterion", _(u"Country")],
    ["lfs.criteria.models.HeightCriterion", _(u"Height")],
    ["lfs.criteria.models.LengthCriterion", _(u"Length")],
    ["lfs.criteria.models.WidthCriterion", _(u"Width")],
    ["lfs.criteria.models.WeightCriterion", _(u"Weight")],
    ["lfs.criteria.models.ShippingMethodCriterion", _(u"Shipping Method")],
    ["lfs.criteria.models.PaymentMethodCriterion", _(u"Payment Method")],
]
```

See also:

Concept of criteria, How to add own criteria

LFS_DELETE_IMAGES If this is set to True images on the file system are going to be deleted after an image has been deleted via the management interface, otherwise they are left untouched. This setting is optional, the default value is True.

LFS_DELETE_FILES If this is set to True files on the file system are going to be deleted after an file/attachment has been deleted via the management interface, otherwise they are left untouched. This setting is optional, the default value is True.

LFS_DOCS Base URL to the LFS docs. This is used for the context aware help link within the management interface. Defaults to <http://docs.getlfs.com/en/latest/>.

LFS_LOCALE Sets the locale for the shop, which is the base for number formatting and the displayed currency. If you don't set it, the current locale of your Python is not touched at all. Example:

```
LFS_LOCALE = "en_US.UTF-8"
```

See also:

<http://en.wikipedia.org/wiki/Locale>, <http://docs.python.org/library/locale.html>

LFS_LOG_FILE Absolute path to LFS' log file.

LFS_RECENT_PRODUCTS_LIMIT The amount of recent products which are displayed within the recent products portlet, e.g. 3.

Orders

LFS_EXTRA_ORDER_STATES Extra states for orders, eg. if you need to mark order as Delivered or such. Should be list of tuples, each containing id and label. Note that id should start from high number (20 or higher) to avoid conflicts if some new ORDER_STATES are added to LFS core. Example value might be: [(20, _('Delivered'))]

There is a signal: `order_state_changed` that is sent when order state was changed and can be used for some special processing.

```
def handle_order_state_changed(sender, order, request, old_state, **kwargs):
    pass
order_state_changed.connect(handle_order_state_changed)
```

LFS_ONE_PAGE_CHECKOUT_FORM The form which is used at checkout page. This setting is optional, the default value is `lfs.checkout.forms.OnePageCheckoutForm`.

Addresses

LFS_AUTO_UPDATE_DEFAULT_ADDRESSES If True then default shipping and invoice addresses (set by customer in his account settings) are automatically updated to the values from last order. Else, default addresses are untouched and are used as defaults in every new order. This setting is optional, the default value is True.

LFS_CHECKOUT_NOT_REQUIRED_ADDRESS During checkout it is possible to not fill in one of the addresses - it will be then copied from another one. By default Shipping address is same as Invoice address, but it can be changed with this setting. Possible values are: `shipping` and `invoice`. Default is `shipping`. Note that you'll have to manually change checkout page template and `lfs.js` if you change it to `invoice`.

By default `checkout_form` (used at `one_page_checkout.html`) has method: `no_address_field` that will return either `no_shipping` or `no_invoice` field, depending on this setting.

Plugins

LFS_ADDRESS_MODEL The model which is used to store addresses. This setting is optional, the default value is `lfs.addresses.models.Address`.

LFS_INVOICE_ADDRESS_FORM The form which is used for shipping addresses. This setting is optional, the default value is `lfs.addresses.forms.InvoiceAddressForm`.

LFS_SHIPPING_ADDRESS_FORM The form which is used for shipping addresses. This setting is optional, the default value is `lfs.addresses.forms.ShippingAddressForm`.

See also:

[How to add own addresses](#)

Required fields

LFS_INVOICE_COMPANY_NAME_REQUIRED If True the company name of the invoice address is required. This setting is optional, the default value is `False`.

LFS_INVOICE_EMAIL_REQUIRED If True the e-mail of the shipping address is required. This setting is optional, the default value is `True`.

LFS_INVOICE_PHONE_REQUIRED If True the phone of the invoice address is required. This setting is optional, the default value is `True`.

LFS_SHIPPING_COMPANY_NAME_REQUIRED If True the company name of the shipping address is required. This setting is optional, the default value is `False`.

LFS_SHIPPING_PHONE_REQUIRED If True the phone of the shipping address is required. This setting is optional, the default value is `False`.

LFS_SHIPPING_EMAIL_REQUIRED If True the e-mail of the shipping address is required. This setting is optional, the default value is `False`.

Units

LFS_UNITS A list of available units for the product.

LFS_PRICE_UNITS A list of available units for the product price.

LFS_BASE_PRICE_UNITS A list of available units for the product base price.

LFS_PACKING_UNITS A list of available units for the product packaging.

E-Mails

LFS_SEND_ORDER_MAIL_ON_CHECKOUT If true, an e-mail with the order details is send to the customer after customer completes checkout screen.

LFS_SEND_ORDER_MAIL_ON_PAYMENT If true, an e-mail is send to the customer after the customer successfully pays for an order

Reviews

REVIEWS_SHOW_PREVIEW True or False. If True the user will see a preview of his review.

REVIEWS_IS_NAME_REQUIRED True or False. If True the name of the review is required.

REVIEWS_IS_EMAIL_REQUIRED True or False. If True the name of the e-mail is required.

REVIEWS_IS_MODERATED True or False. If True the review must be moderated and published before it is public.

PayPal

PAYPAL_RECEIVER_EMAIL Your PayPal id, e.g. info@getlfs.com.

PAYPAL_IDENTITY_TOKEN PayPal's PDT identity token.

LFS_PAYPAL_REDIRECT True or False. If True the customer is automatically redirected to PayPal after he submitted his order. If False the thank-you page is displayed with a link to PayPal.

Management commands

LFS provides few management methods that should be used to keep your LFS instance clean. These are:

LFS Cleanup

Calls all LFS cleanup commands at once. Detailed definition of other cleanup commands can be found below.

```
$ bin/django lfs_cleanup
```

Cleanup customers

LFS creates Customer object for each customer that adds something to the cart. Sometimes these customers do not proceed to the checkout so these Customer objects become useless. This command removes unnecessary customer objects from database. You can run in with cron.daily.

```
$ bin/django cleanup_customers
```

Cleanup addresses

LFS creates Address objects in a number of places. These are bound to the customer or to the order. Sometimes it is possible that Address objects become orphans (eg. when Customer object is removed by not using cleanup_customers command). This command removes unnecessary address objects.

```
$ bin/django cleanup_addresses
```

Cleanup carts

LFS creates Cart objects when customers need to add something to the cart. These objects become unnecessary at some point and it makes no sense to hold them forever. This command removes old carts

```
$ bin/django cleanup_carts
```

How-tos

How to add own addresses

In this how-to you will learn how to add own addresses to LFS, or better yet, how to adapt the address fields to your needs.

You can download the [example application here](#).

Create an Application

First you need to create a default Django application (or use an existing one), where you can put in your plugin. If you do not know how to do this, please refer to the excellent [Django tutorial](#).

Implement the Address Model

The main part of the application consists of a address model which must inherit from the `BaseAddress` class.

```
from lfs.addresses.models import BaseAddress

class MyAddress(BaseAddress):
    values_before_postal = ("firstname+", "lastname+", "company_name")
    values_after_postal = ("phone", "email")

    company_name = models.CharField(_("Company name"), max_length=50, blank=True,
    ↪null=True)
    phone = models.CharField(_("Phone"), blank=True, max_length=20)
    mobile = models.CharField(_("Mobile"), blank=True, max_length=20, required=False)
    email = models.EmailField(_("E-Mail"), blank=True, null=True, max_length=50)
```

In this case we add an extra field `mobile` to LFS' default address. In generell you can add as many fields as you need.

The only LFS specific thing you have to take care of are two attributes: `values_before_postal` and `values_after_postal`, which define the fields, which are displayed before and after the postal address fields. This is used when the address is displayed within LFS, e.g. within the orders or the customer management interface. If you add an `+` at the end of the attribute there will be no `div` around the value. In this ways you can display fields in one row.

Implement the Address Form

Another important part is to add the form, which is displayed to the shop customer when he needs to enter his address. To makes things eaysier you should inherit from `AddressBaseForm`

```
# django imports
from django import forms

# lfs imports
from lfs.addresses.models import AddressBaseForm

# my_addresses imports
from my_addresses.models import MyAddress

class MyAddressForm(AddressBaseForm):
```

```
fields_before_postal = ("firstname", "lastname", "company_name")
fields_after_postal = ("phone", "mobile", "email")

class Meta(AddressForm.Meta):
    model = MyAddress
```

The only LFS specific thing you have to take care of are two attributes: `fields_before_postal` and `fields_after_postal` which define the fields which are displayed before and after the postal address fields. This is used when the form is displayed within LFS, e.g. within the checkout page or within the `my addresses` section.

Plug in the Components

Now, as the code is ready, you can easily plug in your own address:

1. Add your application to the `PYTHONPATH`
2. Add your application to `settings.INSTALLED_APPS` (before `lfs_theme` if you overwrite the default templates):

```
INSTALLED_APPS = (
    ...
    "my_addresses",
)
```

3. Add the model to the `LFS_ADDRESS_MODEL` setting:

```
LFS_ADDRESS_MODEL = "my_addresses.models.MyAddress"
```

4. Add the forms to the `LFS_INVOICE_ADDRESS_FORM` and `LFS_SHIPPING_ADDRESS_FORM` setting:

```
LFS_INVOICE_ADDRESS_FORM = "my_addresses.forms.MyInvoiceAddressForm"
LFS_SHIPPING_ADDRESS_FORM = "my_addresses.forms.MyShippingAddressForm"
```

5. As the address is a new model, you have to synchronize your database:

```
$ bin/django syncdb
```

6. Restart your instance and the address should be displayed to the shop users for instance within the checkout page.

Good to Know

- You can provide different templates to render the addresses. By default LFS tries try to get the specific template (`address_view.html`). If it doesn't exist, it tries to get one of the specific templates (`invoice_address_view.html` or `shipping_address_view.html`).
- You can provide different templates to render the address forms. By default LFS tries try to get the specific template (`address_form.html`). If it doesn't exist, it tries to get one of the specific templates (`invoice_address_form.html` or `shipping_address_form.html`).
- By default LFS automatically updates default addresses to the values from last order. It is possible to change this behavior by setting `LFS_AUTO_UPDATE_DEFAULT_ADDRESSES` to `False`.

See Also

- *Address Settings*

How to add own templates

In this how-to you will learn how to add your own templates for categories and products.

Generally

The content of products and categories are rendered by templates. LFS ships with several default templates and you can add your own.

All registered templates for categories can be selected within the **View** tab of the *Category Management Interface*. All registered templates for products can be selected within the **Data** tab of the *Product Management interface*.

Please refer to the default templates in order to find out which information are provided within the templates. You can also add your customer template tags in order to provide more functionality.

Categories

In order to add a new template for categories go to `lfs.catalog.settings` and add tuple to `CATEGORY_TEMPLATES`.

Example 1

```
(0, {"file": "%s/%s" % (CAT_PRODUCT_PATH, "default.html"),
    "image": IMAGES_PATH + " /product_default.png",
    "name" : _(u"Category with products"),
    }),
```

Which means:

0: The unique id of the category template.

file: The absolute path to the template. `CAT_PRODUCT_PATH` means this is a template which displays the products of a category.

image: The absolute path to the preview image (not used anymore).

name: The pretty name of the template, which is displayed within the template select box.

Example 2

```
(1, {"file": "%s/%s" % (CAT_CATEGORY_PATH, "default.html"),
    "image": IMAGES_PATH + "/category_square.png",
    "name": _(u"Category with subcategories"),
    }),
```

Which means:

1: Unique id of the category template.

file: The absolute path to the template. `CAT_PRODUCT_PATH` means this is a template which displays the sub categories of a category.

image: The absolute path to the preview image (Not used anymore).

name: The pretty name of the template, which is displayed within the template select box.

Products

In order to add a new template for products go to `lfs.catalog.settings` and add tuple to `PRODUCT_TEMPLATES`.

Example

```
(0, {"file" : "%s/%s" % (PRODUCT_PATH, "product_inline.html"),
     "image" : IMAGES_PATH + "/product_default.png",
     "name" : _(u"Default template")}),
```

Which means:

0: The unique id of the product template.

file: The absolute path to the template.

image: The absolute path to the preview image (Not used anymore).

name The pretty name of the template, which is displayed within the template select box.

How to add an own payment processor

In this how-to you will learn how to add a own payment processor.

Create an application

First you need to create a default Django application (or use an existing one), where you can put in your plugin. If you do not know how to do this, please refer to the excellent [Django tutorial](#).

Implement the `PaymentMethodProcessor` class

The main part of the plugin consists of a class which must provide a certain API.

Create the class

Create a class which inherits from `lfs.plugins.PaymentMethodProcessor`:

```
from lfs.plugins import PaymentMethodProcessor

class MyPaymentMethodProcessor(PaymentMethodProcessor):
    pass
```

Add the process method

```
def process(self):
    total = self.order.price
    return {
        "accepted": True,
        "next_url": "http://www.acme.com/payment?id=4711&total=%s" % total,
    }
```

This method is called from LFS when the shop customer submits the checkout page (while selected this payment method). Within that method you can do whatever it is necessary to process your payment, e.g.:

- Call an API via HTTP(S)
- Redirect to another URL

Return Value

The `process` method must return a dictionary with following keys (most of them are optional):

accepted (mandatory) Indicates whether the payment is accepted or not. If this is `False` the customer keeps on the checkout page and gets the message below. If this is `True` the customer will be redirected to `next_url` or to LFS' thank-you page

message (optional) This message is displayed on the checkout page, when the order is not accepted.

message_location (optional) The location, where the message is displayed.

next_url (optional) The url to which the user is redirect after the payment has been processed. If this is not given the customer is redirected to the default thank-you page.

order_state (optional) The state in which the order should be set. It's just PAID. If it's not given the state keeps in SUBMITTED.

Add the get_create_order_time method

```
from lfs.plugins import PM_ORDER_IMMEDIATELY

def get_create_order_time(self):
    return PM_ORDER_ACCEPTED
```

This method is called from LFS to determine when the order is to be created and must return one of following values:

PM_ORDER_IMMEDIATELY The order is created immediately before the payment is processed.

PM_ORDER_ACCEPTED The order is created when the payment has been processed and accepted.

Add the get_pay_link method

In order to provide a link to the customer to re-visit the payment provider and pay his order (if something did go wrong) LFS calls `get_pay_link` method or your class (this is optional).

```
def get_pay_link(self):
    return "http://www.acme.com/payment?id=4711&total=%s" % total
```

The complete plugin

Following all pieces are stucked together to the complete plugin:

```
from lfs.plugins import PaymentMethodProcessor
from lfs.plugins import PM_ORDER_IMMEDIATELY

class ACMEPaymentMethodProcessor(PaymentMethodProcessor):
    """
    Implements the ACME payment processor.
    """
    def process(self):
        return {
            "accepted": True,
            "next_url": self.get_pay_link(),
        }

    def get_create_order_time(self):
        return PM_ORDER_IMMEDIATELY

    def get_pay_link(self):
        total = self.order.price
        return "http://www.acme.com/payment?id=4711&total=%s" % total
```

In this example the order is created immediately and the customer is redirected to the ACME page in order to pay his order. After he has paid he might be redirected to the thank-you page of LFS, but this is completely up to ACME. However, if something goes wrong while he is paying he can always go back to ACME to pay his order because he gets the pay link via the order confirmation mail.

Plug in your payment method

Now as the code is ready, you can easily plugin your payment method:

1. Add your application to the PYTHONPATH.
2. Add the class to the `LFS_PAYMENT_METHOD_PROCESSORS` setting.
3. If your are using models (which is completely up to you), add the application to settings.INSTALLED_APPS and sync your database.
4. *Add a new payment method* and select your payment method within the module field.
5. Select the type of your payment method. Following types are provided:
 - Plain - no further fields are displayed.
 - Bank - fields to enter a bank account are displayed.
 - Credit Card - fields to enter a credit card are displayed.
6. Save the payment method.

Further hints

- Within the `PaymentMethodProcessor` request, the current order or the current cart are available as instance variables:

```
self.request
self.cart (only when get_create_order_time returns PM_ORDER_ACCEPTED)
self.order (only when get_create_order_time returns PM_ORDER_IMMEDIATELY)
```

- When an external payment processor redirects to LFS the current order is still in the session. This means you can redirect to an own view and set the order state to PAID, for instance:

```
from django.core.urlresolvers import reverse
from django.http import HttpResponseRedirect
from lfs.plugins import PAID

def acme_callback_success_view(request):
    order = request.session.get("order")
    order.state = PAID
    order.save()

    return HttpResponseRedirect(reverse("lfs_thank_you"))
```

- All fields of the checkout form are available within the process method via the request variable, e.g.:

```
request.POST.get("invoice_firstname")
```

See also

- *PaymentMethodProcessor API*

How to add custom product pricing module

In this tutorial you will learn how to your own custom product pricing module.

Create an application

First you need to create a default Django application (or use an existing one). If you do not know how to do this, please refer to the excellent [Django tutorial](#).

Implement the price calculator class

Within `__init__.py` file of your application (or anywhere you choose) create a class that inherits from `lfs.plugins.PricingCalculator` and implement all inherited methods.

```
from lfs.plugins import PricingCalculator

class CustomPriceCalculator(PricingCalculator):

    def get_price(self, with_properties=True):
        return self.get_price_net()

    ... Other Methods...
```

Plug in the custom price calculator

1. Add your application to the PYTHONPATH.
2. Add the application to settings.INSTALLED_APPS.
3. If you are using models (which is completely up to you), sync your database.
4. Edit the dictionary lfs.core.settings.LFS_PRICE_CALCULATORS to make your custom pricing calculator available to products in the manage interface

```
LFS_PRICE_CALCULATORS = [  
    ["lfs.gross_price.calculator.GrossPriceCalculator", _(u"Price includes tax")],  
    ["lfs.net_price.calculator.NetPriceCalculator", _(u"Price excludes tax")],  
    ["mycustom_price.CustomPriceCalculator", _(u"My Pricing Calculator")],  
]
```

Set the shop default price calculator

1. Go to the LFS Management Interface.
2. Select Shop / Preferences.
3. Select Default Values and go the Price Calculator section.
4. Select your new pricing calculator from the drop down menu of choices.
5. Save the default values.

Note: All products with an unset price calculator will default to using the shop price calculator.

Set the product pricing calculator

1. Browse to <http://yourshopdomain/manage> and login.
2. Select Catalog / Product.
3. Select the product whose price calculator you wish to change.
4. Select the Data Tab and scroll to the Prices section.
5. Select your new pricing calculator from the drop down menu.
6. Click on Save Data.
7. Now browse to the customer view of the product and you should see the price as calculated by your custom pricing calculator.

How to use localized addresses

Address localization is turned on by default in LFS. To turn off Address l10n in settings.py set:

```
POSTAL_ADDRESS_L10N = False
```

Customize address labels and requirement

If you wish to customize the address labels and whether the address line is required or not, you can add the following variables to settings.py:

```
POSTAL_ADDRESS_LINE1, POSTAL_ADDRESS_LINE2, POSTAL_ADDRESS_CITY,
POSTAL_ADDRESS_STATE, POSTAL_ADDRESS_CODE
```

Each of these variables is set to a tuple of the format:

```
('label', True/False)
```

label is used to label the field, and the second boolean value sets whether the field is required or not, e.g.:

```
POSTAL_ADDRESS_LINE1 = ("Department", True)
```

How to create a product export script

Overview

LFS provides a generic export engine for products ([see here for more](#)). In this tutorial you will learn how to create your own scripts to format the data like you want to.

Create an application

First you need to create a default Django application (or use an existing one). If you do not know how to do this, please refer to the excellent [Django tutorial](#).

Create the code

Within the `__init__.py` of your application create a function that will return the products, like so:

```
1  # python imports
2  import csv
3
4  # django imports
5  from django.http import HttpResponseRedirect
6  from django.core.mail import send_mail
7
8  # lfs imports
9  from lfs.export.utils import register
10
11 def export(request, export):
12     """Export method for acme.com
13     """
14     response = HttpResponseRedirect(mimetype="text/csv")
15     response["Content-Disposition"] = "attachment; filename=acme.txt"
16
17     writer = csv.writer(response, delimiter=";", quotechar='"', quoting=csv.QUOTE_ALL)
18
19     for product in export.get_products():
20         writer.writerow((product.id, product.get_name()))
21
22     return response
```

```
23  
24 register(export, "acme.com")
```

The code explained

- 1-6** Some simple Python and Django imports. These may vary for your code.
- 9** Imports the register method to register your script.
- 11** The function which will contain your code to create the exported data.
- 14/15** We decided to give a response with a download back. Your code may vary here.
- 16** We decided to use Python csv modules. Your code may vary here.
- 19** All selected products can be get with the `get_products` method of the passed export object.
- 24** The registration of your function. This line must be called while Django is starting up.

Plug in the export script

Now go `Utils / Export` within the LFS Management Interface, create a new export, select the products and your newly script and call it via the `Export` button. (*See here for more*).

How to add own action groups

Overview

In this how-to you will learn how to add new `Actions Groups` and how to use them within your templates.

Add a new action group

1. Login as admin.
2. Go to Django's admin interface:

```
http://localhost:8000/admin/
```
3. Go to `Core / Action groups`.
4. Click on the `Add Action Group` button.
5. Enter the name of the group, e.g. `Sidebar`.

Display the actions within your templates

Insert the `actions` tag to your template as following:

```
{% actions Sidebar %}  
{% for action in actions %}  
    <div>  
        <a href="{{ action.link }}">  
            {{ action.title }}  
        </a>
```

```
</div>
{% endfor %}
```

Note: We pass the above given group name to the `actions` tab. In this case `Sidebar`.

Add actions to the group

1. Open the LMI and go to Shop / Actions.
2. Click on the Add Action button.
3. Fill in the provided form and select your new group.
4. Click on the Save Action button.

See also

- *[General about actions](#)*
- *[Actions management interace](#)*

How to create a theme

In this how-to you will learn how to create a theme for LFS.

Note: You can download the whole theme [here](#).

Preparations

First you have to create a new Django application. This is beyond the purpose of this tutorial and you should refer to [Django's excellent tutorial](#) if you want to learn more.

In short, your starting file structure should look like this:

```
mytheme
  __init__.py
  templates
    lfs
```

Registration

Register mytheme to Django's template engine.

1. Move the mytheme folder to the PYTHONPATH.
The easiest way to do that is to put it into the `lfs_project` folder of the buildout.
2. Register the theme
Add mytheme to `INSTALLED_APPS` **before** `lfs`theme:

```
INSTALLED_APPS = (  
    ...  
    "mytheme",  
    "lfs",  
    "django.contrib.admin",  
    ...  
)
```

Copy templates

Now copy the templates you want to change into the lfs folder of mytheme and adapt them to your needs.

Important: you have to keep the original path, e.g: base.html must be within the root of the lfs folder whereas the cart portlet (cart.html) must be within the portlets folder:

```
mytheme  
  __init__.py  
  templates  
    lfs  
      base.html  
      portlets  
        cart.html
```

Use own css

To use own CSS several steps are necessary.

1. Create a `static` folder within mytheme:

```
mytheme  
  static  
  ...
```

2. Within that create a new CSS-file, e.g. mytheme.css and add your CSS rules, e.g.:

```
.breadcrumbs li {  
    color: red !important;  
}
```

Alternatively you might copy main.css from lfstheme and adapt it to your needs.

3. Go to the `lfs_project/media` folder and create a symbolic link to the static folder:

```
$ ln -s <path/to/buildout>/lfs_project/mytheme/static mytheme
```

4. Copy base.html to mytheme/templates/lfs (if you haven't done it so far)
5. Include your CSS file to the header:

```
<link rel="stylesheet" type="text/css" href="{% static 'mytheme/mytheme.css' %}">
```

6. Optionally delete the link to main.css (if you just want to use your own CSS).

How to add an own order numbers generator

Overview

LFS order numbers generator is pluggable. In this tutorial you will learn how to add an own one.

Please see also the [complete example application](#) or refer to the default implementation of LFS within `lfs_order_numbers`.

Create an application

First you need to create a default Django application (or use an existing one). If you do not know how to do this, please refer to the excellent [Django tutorial](#).

The structure of the application should look at least like this:

```
my_order_numbers
    __init__.py
    models.py
```

Implement the application

Add the model

Within `models.py` file of your application create a class called `OrderNumberGenerator` which inherits from LFS' `OrderNumberGenerator` base class and add a method to it called `get_next`:

```
from lfs.plugins import OrderNumberGenerator as Base

class OrderNumberGenerator(Base):
    def get_next(self, formatted=True):
        return "DOE-4711"
```

The `get_next` method is called when the shop customer submits a new order. It **must** return a character value which will become the order number of the new order.

Plug in your order number generator

Now as the code is ready, you can easily plugin your payment method:

1. Add your application to the `PYTHONPATH`.
2. Add your class to `settings.py`

```
LFS_ORDER_NUMBER_GENERATOR = "my_order_numbers.models.OrderNumberGenerator"
```
3. Add your application to `settings.INSTALLED_APPS` and sync the database.

And that's it

You should now see your form within the `Order Numbers` tab within `Shop/Preference` and the `get_next` method of your model should be called to generate a new order number.

Optionally Add your own Template

Optionally you can add your own HTML for the management interface. For this just add the `order_numbers_tab.html` template to your application:

```
my_order_numbers
  templates
    manage
      order_numbers
        order_numbers_tab.html
```

Please refer to the standard template of LFS to get more details. You can find this on following place:

```
lfs/templates/manage/order_numbers/order_numbers_tab.html
```

In this case please make sure that your `my_order_numbers` application stands **before** `lfs` within `INSTALLED_APPS` of `settings.py` so that LFS' default `order_numbers_tab.html` template is overwritten.

Further hints

- The form is automatically created from the model you provide. However you can provide a own own by overwriting the `get_form` method. *See the API for more.*
- If you just want to exclude certain fields from the automatically generated form you can overwrite the `exclude_form_fields`. The return value is just passed to the `exclude` attribute of the from Meta class. *See the API for more.*

Available information

Within the `get_next` method of your new class you have access to following information:

self.request The current request

self.user The current user

self.customer The current customer

self.cart The current cart

self.order The order which is about to be created.

Please note that you have also access to the products of the order via the `items` attribute. For instance:

```
for item in self.order.items.all():
    product = item.product
```

See the also the `Order` and `OrderItem` classes for more information.

How to add an own shipping price calculator

In this how to you will learn how to add your own shipping price calculator.

Create an application

First you need to create a default Django application (or use an existing one), where you can put in your plugin. If you do not know how to do this, please refer to the excellent [Django tutorial](#).

Implement the `ShippingMethodPriceCalculator` class

The main part of the plugin consists of a class which must provide a certain API.

Create the class

Create a class which inherits from `lfs.plugins.ShippingMethodPriceCalculator`:

```
from lfs.plugins import ShippingMethodPriceCalculator

class MyShippingMethodPriceCalculator(ShippingMethodPriceCalculator):
    pass
```

Add the `get_price_net` and `get_price_gross` methods

This methods are called from LFS to display the shipping price or to calculate the total price of the cart or order.

```
def get_price_net(self):
    # This doesn't make much sense, but the net price is always 11.0
    return 11.0

def get_price_gross(self):
    return 11.0 * ((100 + self.shipping_method.tax.rate) / 100)
```

The complete plugin

Following all pieces are stucked together to the complete plugin:

```
from lfs.plugins import ShippingMethodPriceCalculator

class MyShippingMethodPriceCalculator(ShippingMethodPriceCalculator):
    def get_price_net(self):
        # This doesn't make much sense, but the net price is always 11.0
        return 11.0

    def get_price_gross(self):
        return 11.0 * ((100 + self.shipping_method.tax.rate) / 100)
```

Plug in your payment method

Now as the code is ready, you can easily plugin your shipping method price calculator:

1. Add your application to the `PYTHONPATH`.
2. Add the class to the `LFS_SHIPPING_METHOD_PRICE_CALCULATORS` setting.
3. If your are using models (which is completely up to you), add the application to `settings.INSTALLED_APPS` and sync your database.
4. Add a new shipping method and select your price calculator within the `price_calculator` field.
5. Save the shipping method.

Further hints

- Within the `ShippingMethodPriceCalculator` the current request and the shipping method are available as instance variables:

```
self.request
self.shipping_method
```

- With the `request` you have access to the current cart (in case you need it):

```
from lfs.cart.utils import get_cart
cart = get_cart(self.request)
```

- With the `request` you have access to the current customer (in case you need it):

```
from lfs.customer.utils import get_customer
customer = get_customer(self.request)
```

How to integrate localized version of TinyMCE

LFS ships with main package of TinyMCE that doesn't contain any translation files. It is easy to use internationalized version of TinyMCE by adding few things into your theme.

Steps

1. Download TinyMCE

Go to TinyMCE website and [download TinyMCE x.x jQuery package](#)

2. Extract TinyMCE into your theme

Extract downloaded TinyMCE into your *Theme*, e.g. `theme/static/manage/tiny_mce_x_x/`

3. Download TinyMCE language file(s)

Go to TinyMCE website and [download language file\(s\)](#) that you need

4. Extract TinyMCE language files into your theme

Go to folder where you've just extracted main package of TinyMCE, e.g.: `theme/static/manage/tiny_mce_x_x/` and extract language files into proper directories.

5. Copy `manage_base.html` to your theme

Copy `lfs/templates/manage/manage_base.html` to your theme: `mytheme/templates/manage/manage_base.html`

6. Copy `lfs_tinymce.js` to your theme

Copy `lfs/static/js/lfs_tinymce.js` to your theme: `mytheme/static/js/lfs_tinymce.js` (if you use different path then you have to update it at `manage_base.html` in step 7)

7. Modify `lfs_tinymce.js` (copy located at your theme)

Change/add highlighted parts of TinyMCE initialization script:

```
// Theme options
$(selector).tinymce({
    // Location of TinyMCE script
    script_url : '/static/manage/tiny_mce_x_x/tiny_mce.js',
```

```
// General options
theme : "advanced",
plugins : "safari,save,iespell,directionality,fullscreen,xhtmlxtras,media",

theme_advanced_buttons1 : buttons,
theme_advanced_buttons2 : "",
theme_advanced_buttons3 : "",
theme_advanced_buttons4 : "",
theme_advanced_toolbar_location : "top",
theme_advanced_toolbar_align : "left",
save_onsavecallback : "save",
relative_urls : false,
cleanup : false,
height : height,
language : LFS_LANGUAGE,
content_css : "/static/css/tinymce_styles.css",
setup : function(ed) {
    ed.addButton('image', {
        onclick : function(e) {
            imagebrowser(e, ed);
        }
    });
}
});
```

8. Customize manage_base.html at your theme

Replace:

```
<script type="text/javascript" src="{% static 'tiny_mce-3.4.2/jquery.tinymce.js'
↪%}"></script>
```

with (use path to TinyMCE folder that you created in step 2):

```
<script type="text/javascript" src="{% static 'manage/tiny_mce_x_x/jquery.tinymce.
↪js' %}"></script>
```

Add following code to <head> section:

```
<script type="text/javascript">
    var LFS_LANGUAGE = '{{ LANGUAGE_CODE|lower }}';
</script>
```

Note that for some languages LANGUAGE_CODE used by Django may differ from language code used by TinyMCE. For such cases you'll probably have to write your own tag/filter that will map Django's language code to TinyMCE's language code (or you'll just hard code it).

How to add own criteria

In this how-to you will learn how to add own criteria to LFS.

The goal in this how-to is to create a criterion, for which the shop manager can enter a SKU and decide (via operators) whether the criterion is valid if the product with the entered SKU is within the cart or not.

You can download the [example application](#) here.

Create an application

First you need to create a default Django application (or use an existing one), where you can put in your plugin. If you do not know how to do this, please refer to the excellent [Django tutorial](#).

Implement the Criterion Model

The main part of the application consists of a criterion model which must inherit from the `Criterion` base class.

Create the Class

```
class ProductCriterion(Criterion):
    value = models.CharField(max_length=100)
```

The only attribute we need is the value the shop manager will save for the criterion. The attribute can have any type you need. In this example we use a simple character field. The entered SKU is checked within the products in the cart. Dependent on the chosen operator the criteria is valid if the product is within the cart or not.

Implement necessary Methods

In the next steps we implement all necessary methods which are needed to make the criterion work. In this case these are `get_operators` and `is_valid`.

The `get_operators` method needs to return the available operators for this criterion. It is a list of list, whereas the first value is an integer and the second value is the name of the operator.

```
def get_operators(self):
    return [
        [0, _(u"Is in cart")],
        [1, _(u"Is not in cart")],
    ]
```

The `is_valid` method needs to return a boolean. If it returns `True` the criterion is considered valid, it returns `False` the criterion is considered not valid.

```
def is_valid(self):
    if self.product:
        return self.value == self.product.sku
    elif self.cart:
        result = any([self.value == item.product.sku for item in self.cart.get_
↪items()])
        return result if self.operator == 0 else not result
    else:
        return False
```

Note: Within the `is_valid` method (as in all methods of the `Criterion` class) following attributes are available:

product This is only set, when the criterion is called from the product detail view otherwise it is `None`.

cart The current cart of the current user.

request The current request.

Plug in the Criterion

Now as the code is ready, you can easily plugin your own criterion:

1. Add your application to the PYTHONPATH
2. Add your application to settings.INSTALLED_APPS and sync your database:

```
INSTALLED_APPS = (
    ...
    "product_criterion",
)
```

3. Add the class to the *LFS_CRITERIA* setting:

```
LFS_CRITERIA = [
    ...
    ["product_criterion.models.ProductCriterion", _("Product Criterion")],
]
```

4. As all criteria are models, you have to synchronize your database:

```
$ bin/django syncdb
```

5. Restart your instance and the criterion should be available for selection, for instance within the discount criteria tab.

And that's it

You should now see your criterion within the criteria tab of Discounts for instance. You can enter a product SKU to it and select one of the above mentioned operators.

Good to know

- You can also create criteria with select or multiple select fields. See the *API* or the default Country criterion within `lfs.criteria.models` for more.
- You can override more than the two mentioned methods above. See the *Criterion API* which methods are provided by the base class.

See Also

- *Criteria concept*
- *Criterion API*
- Look into the default criteria within `lfs.criteria.models` to see how these are implemented

CHAPTER 7

Miscellaneous Information

CHAPTER 8

Indices and Tables

- glossary
- genindex
- search

A

Action, [28, 33](#)
Average Rating Portlet, [26](#)

B

Benchmarking LFS, [91](#)

C

Cart, [61](#)
Cart Portlet, [26](#)
Categories Portlet, [26](#)
Category, [16, 42](#)
Changing models, [92](#)
Criteria, [17](#)
Customer, [58](#)
Customer Tax, [41](#)

D

Delivery time, [29, 34](#)
Delivery Time Portlet, [26](#)
Developing LFS, [89](#)
Discount, [68](#)
Downloadable
 File, [56](#)

E

exclude_form_fields() (lfs.plugins.OrderNumberGenerator
 method), [96](#)

F

Featured, [65](#)
Featured Products Portlet, [27](#)
File
 Downloadable, [56](#)
Filter Portlet, [27](#)
For Sale Portlet, [27](#)

G

get_base_packing_price() (lfs.plugins.PriceCalculator
 method), [98](#)

get_base_packing_price_gross()
 (lfs.plugins.PriceCalculator method), [98](#)
get_base_packing_price_net()
 (lfs.plugins.PriceCalculator method), [98](#)
get_base_price() (lfs.plugins.PriceCalculator method), [97](#)
get_base_price_gross() (lfs.plugins.PriceCalculator
 method), [98](#)
get_base_price_net() (lfs.plugins.PriceCalculator
 method), [98](#)
get_create_order_time() (lfs.plugins.PaymentMethodProcessor
 method), [97](#)
get_customer_tax() (lfs.plugins.PriceCalculator method),
 [98](#)
get_customer_tax_rate() (lfs.plugins.PriceCalculator
 method), [98](#)
get_effective_price() (lfs.plugins.PriceCalculator
 method), [98](#)
get_for_sale_price() (lfs.plugins.PriceCalculator
 method), [98](#)
get_for_sale_price_gross() (lfs.plugins.PriceCalculator
 method), [99](#)
get_for_sale_price_net() (lfs.plugins.PriceCalculator
 method), [98](#)
get_form() (lfs.plugins.OrderNumberGenerator method),
 [96](#)
get_next() (lfs.plugins.OrderNumberGenerator method),
 [96](#)
get_operators() (lfs.criteria.models.Criterion method), [95](#)
get_pay_link() (lfs.plugins.PaymentMethodProcessor
 method), [97](#)
get_price() (lfs.plugins.PriceCalculator method), [99](#)
get_price() (lfs.plugins.ShippingMethodPriceCalculator
 method), [100](#)
get_price_gross() (lfs.plugins.PriceCalculator method),
 [99](#)
get_price_gross() (lfs.plugins.ShippingMethodPriceCalculator
 method), [100](#)
get_price_net() (lfs.plugins.PriceCalculator method), [99](#)
get_price_net() (lfs.plugins.ShippingMethodPriceCalculator
 method), [100](#)

get_product_tax() (lfs.plugins.PriceCalculator method), 100

get_product_tax_rate() (lfs.plugins.PriceCalculator method), 100

get_selectable_values() (lfs.criteria.models.Criterion method), 95

get_standard_price() (lfs.plugins.PriceCalculator method), 99

get_standard_price_gross() (lfs.plugins.PriceCalculator method), 99

get_standard_price_net() (lfs.plugins.PriceCalculator method), 99

get_tax() (lfs.plugins.ShippingMethodPriceCalculator method), 100

get_tax_rate() (lfs.plugins.ShippingMethodPriceCalculator method), 100

get_template() (lfs.criteria.models.Criterion method), 95

get_value() (lfs.criteria.models.Criterion method), 95

get_value_type() (lfs.criteria.models.Criterion method), 95

Global images, 32, 41

Google Analytics, 37

H

HTML, 25, 32, 55, 57

I

Images, 32, 41

init() (lfs.plugins.OrderNumberGenerator method), 96

Installation, 2

Invoice Address, 58

is_valid() (lfs.criteria.models.Criterion method), 95

L

Latest Portlet, 27

lfs.criteria.models.Criterion (built-in class), 94

lfs.plugins.OrderNumberGenerator (built-in class), 96

lfs.plugins.PaymentMethodProcessor (built-in class), 97

lfs.plugins.PriceCalculator (built-in class), 97

lfs.plugins.ShippingMethodPriceCalculator (built-in class), 100

M

Management commands, 104

Manufacturers, 17

Marketing, 30

O

Order, 59

Order Numbers, 39

Orders, 59

Overview, 1

P

Page, 32, 55

Page Portlet, 27

Payment Method, 30

Payment method, 36, 79

Portlet

- Average Rating, 26
- Cart, 26
- Categories, 26
- Delivery Time, 26
- Featured Product, 27
- Filter, 27
- For Sale, 27
- Latest, 27
- Page, 27
- Recent, 28
- Related, 28
- Text, 28
- Top Seller, 28

Portlets, 25, 39, 51

Price Calculator, 37, 44, 52, 111

price_includes_tax() (lfs.plugins.PriceCalculator method), 100

process() (lfs.plugins.PaymentMethodProcessor method), 97

Product, 15, 44, 52

- Configurable, 16
- Default, 15
- Variant, 16
- with Variants, 16

Property, 22, 24, 54

Property Group, 53

R

Recent Portlet, 28

Related Portlet, 28

render() (lfs.criteria.models.Criterion method), 96

Reviews, 32, 63

- Mails, 70

S

SEO, 39, 51

Settings, 100

Shipping Address, 58

Shipping Method, 39, 82

Shipping Methods, 29

Static Block, 25, 47, 57

T

Tax

- Customer, 41
- Product, 40

Taxes, 24

Template, [47](#)
Testing LFS, [90](#)
Text Portlet, [28](#)
Top Seller Portlet, [28](#)
Topseller, [66](#)

U

update() (lfs.criteria.models.Criterion method), [96](#)
Upgrade notes, [8](#)

V

Voucher, [69](#)
Voucher Group, [69](#)