spreecommerce™
BEHIND THE BEST STOREFRONTS™

Calculators - Developer Guide | Spree Commerce

Overview

Spree makes extensive use of the `Spree::Calculator` model and there are several subclasses provided to deal with various types of calculations (flat rate, percentage discount, sales tax, VAT, etc.) All calculators extend the `Spree::Calculator` class and must provide the following methods:

```
def self.description
  # Human readable description of the calculator
end

def compute(object=nil)
  # Returns the value after performing the required calculation
end
```

Calculators link to a `calculable` object, which are typically one of `Spree::ShippingMethod`, `Spree::TaxRate`, or `Spree::Promotion::Actions::CreateAdjustment`. These three classes use the `Spree::Core::CalculatedAdjustment` module to provide an easy way to calculate adjustments for their objects.

Available Calculators

The following are descriptions of the currently available calculators in Spree. If you would like to add your own, please see the Creating a New Calculator section.

Default Tax

For information about this calculator, please read the Taxation guide.

Flat Percent Per Item Total

This calculator has one preference: `flat_percent` and can be set like this:

```
calculator.preferred_flat_percent = 10
```

This calculator takes an order and calculates an amount using this calculation:

```
[item total] x [flat percentage]
```

For example, if an order had an item total of $31 and the calculator was configured to have a flat percent amount of

10, the discount would be $3.10, because $31 x 10% = $3.10.

Flat Rate

This calculator can be used to provide a flat rate discount.

This calculator has two preferences: `amount` and `currency`. These can be set like this:

```
calculator.preferred_amount = 10
calculator.currency = "USD"
```

The currency for this calculator is used to check to see if a shipping method is available for an order. If an order's currency does not match the shipping method's currency, then that shipping method will not be displayed on the frontend.

This calculator can take any object and will return simply the preferred amount.

Flexi Rate

This calculator is typically used for promotional discounts when you want a specific discount for the first product, and then subsequent discounts for other products, up to a certain amount.

This calculator takes three preferences:

- `first_item`: The discounted price of the first item(s).
- `additional_item`: The discounted price of subsequent items.
- `max_items`: The maximum number of items this discount applies to.

The calculator computes based on this:

[first item discount] + (([items_count*] - 1) x [additional item discount])

- up to the `max_items`

Thus, if you have ten items in your shopping cart, your `first_item` preference is set to $10, your `additional_items` preference is set to $5, and your `max_items` preference is set to 4, the total discount would be $25:

- $10 for the first item
- $5 for each of the 3 subsequent items: $5 * 3 = $15
- $0 for the remaining 6 items

Free Shipping

This calculator will take an object, and then work out the shipping total for that object. Useful for when you want to apply free shipping to an order.

********** TODO **********

This is a little confusing and vague. Need to investigate more and explain better. Also, might this be obsolete with the new split shipments functionality?

****

Per Item

The Per Item calculator computes a value for every item within an order. This is useful for providing a discount for a specific product, without it affecting others.

This calculator takes two preferences:

- `amount`: The amount per item to calculate.
- `currency`: The currency for this calculator.

This calculator depends on its `calculable` responding to a `promotion` method, which should return a `Spree::Promotion` (or similar) object. This object should then return a list of rules, which should respond to a `products` method. This is used to return a result of matching products.

The list of matching products is compared against the line items for the order being calculated. If any of the matching products are included in the order, they are eligible for this calculator. The calculation is this:

[matching product quantity] x [amount]

Every matching product within an order will add to the calculator's total. For example, assuming the calculator has an `amount` of 5 and there's an order with the following line items:

- Product A: $15.00 x 2 (within matching products)
- Product B: $10.00 x 1 (within matching products)
- Product C: $20.00 x 4 (excluded from matching products)

The calculation would be:

```
= (2 x 5) + (1 x 5)
= 10 + 5
```

meaning the calculator will compute an amount of 15.

Percent Per Item

The Percent Per Item calculator works in a near-identical fashion to the Per Item Calculator, with the exception that rather than providing a flat-rate per item, it is a percentage.

Assuming a calculator amount of 10% and an order such as this:

- Product A: $15.00 x 2 (within matching products)
- Product B: $10.00 x 1 (within matching products)
- Product C: $20.00 x 4 (excluded from matching products)

The calculation would be:

```
= ($15 x 2 x 10%) + ($10 x 10%)
= ($30 x 10%) + ($10 x 10%)
= $3 + $1
```

The calculator will calculate a discount of $4.

Price Sack

The Price Sack calculator is useful for when you want to provide a discount for an order which is over a certain price. The calculator has four preferences:

- `minimal_amount`: The minimum amount for the line items total to trigger the calculator.
- `discount_amount`: The amount to discount from the order if the line items total is equal to or greater than the `minimal_amount`.

- `normal_amount`: The amount to discount from the order if the line items total is less than the `minimal_amount`.
- `currency`: The currency for this calculator. Defaults to the currency you have set for your store with `Spree::Config[:currency]`

Suppose you have a Price Sack calculator with a `minimal_amount` preference of $50, a `normal_amount` preference of $2, and a `discount_amount` of $5. An order with a line items total of $60 would result in a discount of $5 for the whole order. An order of $20 would result in a discount of $2.

Creating a New Calculator

To create a new calculator for Spree, you need to do two things. The first is to inherit from the `Spree::Calculator` class and define `description` and `compute` methods on that class:

```
class CustomCalculator < Spree::Calculator
  def self.description
    # Human readable description of the calculator
  end

  def compute(object=nil)
    # Returns the value after performing the required calculation
  end
end
```

If you are creating a new calculator for shipping methods, please be aware that you need to inherit from `Spree::ShippingCalculator` instead, and define a `compute_package` method:

```
class CustomCalculator < Spree::ShippingCalculator
  def self.description
    # Human readable description of the calculator
  end

  def compute_package(package)
    # Returns the value after performing the required calculation
  end
end
```

The second thing is to register this calculator as a tax, shipping, or promotion adjustment calculator by calling code like this at the end of `config/initializers/spree.rb` inside your application (`config` variable defined for brevity):

```
config = Rails.application.config
config.spree.calculators.tax_rates << CustomCalculator
config.spree.calculators.shipping_methods << CustomCalculator
config.spree.calculators.promotion_actions_create_adjustments << CustomCalculator
```

For example if your calculator is placed in `app/models/spree/calculator/shipping/my_own_calculator.rb` you should call:

```
config = Rails.application.config
config.spree.calculators.shipping_methods << Spree::Calculator::Shipping::MyOwnCalculator
```

Determining Availability

By default, all shipping method calculators are available at all times. If you wish to make this dependent on something

from the order, you can re-define the `available?` method inside your calculator:

```
class CustomCalculator < Spree::Calculator
  def available?(object)
    object.currency == "USD"
  end
end
```

Calculated Adjustments

If you wish to use Spree's calculator functionality for your own application, you can include the `Spree::Core::CalculatedAdjustments` module into a model of your choosing.

```
class Plan < ActiveRecord::Base
  include Spree::Core::CalculatedAdjustments
end
```

To have calculators available for this class, you will need to register them:

```
config.spree.calculators.plans << CustomCalculator
```

Then you can access these calculators by calling this method:

```
Plan.calculators
```

Using this method, you can then display the calculators as you please. Each object for this new class will need to have a calculator associated so that adjustments can be calculated on them.

This module provides a `has_one` association to a `calculator` object, as well as some convenience helpers for creating and updating adjustments for objects. Assuming that an object has a calculator associated with it first, creating an adjustment is simple:

```
plan.create_adjustment("#{plan.name}", <target object>, <calculable object>)
```

To update this adjustment:

```
plan.update_adjustment(<adjustment object>, <calculable object>)
```

To work out what the calculator would compute an amount to be:

```
plan.compute_amount(<calculable object>)
```

`create_adjustment`, `update_adjustment` and `compute_amount` will call `compute` on the `Calculator` object. This `calculable` amount is whatever object your `CustomCalculator` class supports.

- **Tutorials**
  - Getting Started
  - Extensions
  - Deface Overrides

- **Source Code**
  - About
  - Navigating
  - Getting Help
  - Contributing
- **The Core**
  - Addresses
  - Adjustments
  - Calculators
  - Inventory
  - Orders
  - Payments
  - Preferences
  - Products
  - Promotions
  - Shipments
  - Taxation
- **Customization**
  - Authentication
  - Internationalization (i18n)
  - View
  - Asset
  - Logic
  - Use S3
  - Checkout
- **Deployment**
  - Heroku
  - Shelly Cloud
  - Ansible Ubuntu Deployment
  - Manual Ubuntu Deployment
  - Deployment Options
  - Deployment Tips
  - Requesting/Configuring SSL
- **Advanced Topics**
  - Developer Tips
  - Gateway Specific Information
  - Migrating to Spree
  - Search Engine Optimization
  - Security
  - Testing Spree Applications
- **Upgrade Guides**
  - 0.60.x to 0.70.x
  - 0.70.x to 1.0.x
  - 1.0.x to 1.1.x
  - 1.1.x to 1.2.x

- 1.2.x to 1.3.x
- 1.3.x to 2.0.x
- 2.0.x to 2.1.x
- 2.1.x to 2.2.x
- 2.2.x to 2.3.x

Product

- Platform
- Wombat
- Support
- Privacy Policy
- Terms of Service

Developers

- Overview
- Documentation
- Community
- License

Partners

- Pro Services
- Training
- Partnership Program
- Payments

About Spree Commerce

Spree Commerce is an automated enterprise solution built specifically for ecommerce. We effectively manage your operations so you can focus on serving your customers and growing your business.

Take it for a test drive

You can even create your own personal store.

Try The Demo