Vespolina Documentation

Release

Vepolina community

August 27, 2013

Contents

Vespolina is a php5 ecommerce system that plugs into Symfony2. One of the primary goals is to create a decoupled system, libraries that can be used in more applications then just Symfony applications.

CHAPTER 1

Quick Tour

1.1 Quick Tour

If you want to start learning Vespolina, or want to see what it's all about you can start right her!

1.1.1 Installing the sandbox

The Vespolina Sandbox demo's a lot of the features Vespolina offers. While it's possible to use as a development platform, it's not intended as such.

Requirements

- PHP > 5.4.0 (5.3 might work...)
- MongoDB (at this moment DBAL is not yet fully supported)

Getting the code

The recommended way is to clone the sandbox from: https://github.com/vespolina/vespolina-sandbox.git

Then you can use *composer install* to download the dependencies.

1.1.2 Vespolina Store's

A short introduction to Vespolina stores. What can you do with them and how to use them.

1.1.3 Vespolina Catalog

A short introduction to the Vespolina Catalog. What types of products default are supported.

1.1.4 Vespolina Sales

A short introduction to Vespolina Sales. The different types and options.

1.2 Installing the sandbox

The Vespolina Sandbox demo's a lot of the features Vespolina offers. While it's possible to use as a development platform, it's not intended as such.

1.2.1 Requirements

- PHP > 5.4.0 (5.3 might work...)
- MongoDB (at this moment DBAL is not yet fully supported)

1.2.2 Getting the code

The recommended way is to clone the sandbox from: https://github.com/vespolina/vespolina-sandbox.git

Then you can use *composer install* to download the dependencies.

1.3 Vespolina Store's

A short introduction to Vespolina stores. What can you do with them and how to use them.

1.4 Vespolina Catalog

A short introduction to the Vespolina Catalog. What types of products default are supported.

1.5 Vespolina Sales

A short introduction to Vespolina Sales. The different types and options.

CHAPTER 2

Book

The book contains a better explanation of the concepts of Vespolina. It contains documents that describe how a component works, why it's build as it is and how you could use it.

2.1 Vespolina philosophy

Vespolina aims to deliver a framework that supports a wide variety of (e)commerce flows and processes.

2.2 Vespolina Installation

This chapter will explain in detail how to get a vespolina application up and running.

2.2.1 Installing a distribution

- composer
- archive

2.2.2 Start building your application

The following steps assume you are setting up a webshop. If you are trying to setup something different, like a inventory system, you should find a cookbook entry about it, or read the components documentation.

2.3 Vespolina Store

The Store binds all commerce together in a store. It regulates controllers and processes. And also provides some handlers for common store types like a default webshop, daily deal or a campaign store.

2.3.1 Creating a store

```php
<?php
// Retrieve the manager
$storeManager = $this->getContainer()->get('vespolina.store_manager');
// Create the store
$store = $storeManager->createStore('default_store', 'myshop.com');
// Save the store the persistence
$storeManager->updateStore($store);
```

2.4 Products

Short intro about products

2.5 Taxonomy

Short intro about taxonomy

2.6 Partners

Short intro about customers and other partners.

2.7 Sales

Short intro about sales

2.8 Fulfilment

Short intro about fulfilment

2.9 Process

Short intro about the different processes

CHAPTER 3

Cookbook

The cookbook contains examples how to achive a specific goal. For instance it provides an example to configure a 1-day-deal shop.

No cookbook entries have been written yet.

Possible entries:

- Creating an inventory system
- Creating a POS
- Setting up a B2B process

CHAPTER 4

Components

4.1 Components

This part of the documentation covers the components available in Vespolina. The components are a more low level part of Vespolina. They aren't plug&play pieces of software. But at the same time, they contain the most important pieces of Vespolina.

4.1.1 Core

The Core component is required for almost all other components. It contains most of the `Entities` and there interfaces.

Beside the entities, the Core component also includes some default Exceptions, a Vespolina specific EventDispatcher.

Installing

```
$ composer require vespolina/
```

Entities

The entities have `a seperate section` in the documentation

Exceptions

VespolinaCore defines some global exceptions.

FunctionNotSupportedException

To indicate missing/not-implemented functionality. Not used at the moment

IdentifierCheckDigitException

Thrown when a invalid identifier is given/generated.

InvalidConfigurationException

Not used at the moment

InvalidInterfaceException

Not used at the moment.

InvalidOptionsException

Thrown when creating an order with invalid or missing options.

4.1.2 Cart

The cart component has a basic cart implementation. It handles adding, updating and removing of products. But also has a pricing provider to handle things like taxes.

Installation

```
$ composer require vespolina/cart
```

4.1.3 Product

General Concepts

The Product class is a container for basic product information. This includes product features, product options and identifiers. A Product can be conceptualized in the same way a product in the physical product is understood. A product has certain physical properties that do not change during the course of its life in the commerce process. A product come into a warehouse and it becomes inventory, but the product itself doesn't change. Same with when a product is put on the shelf in a store. It is now merchandise, but the product hasn't changes, just its role in the commerce process has changed.

When a product is moved into a SalesChannel (for example, an ecommerce site) to be purchased, the Product object now becomes part of a Merchandise object. The Merchadise object contains additional information like a description, the price, or addition images that are site specific. Having the the Product become Merchandise also allows a Product to be priced differently or display different desciptions in multiple SalesChannels

Product class

The minimal data needed for a Product is a name.

Features

Features are attributes of a product. The following are features of Kent Beck's Test Driven Development

type	name
Binding	Paperback
Pages	240
Publisher	Addison-Wesley Professional
Language	English

Options

Options are variations on the product. An example of this might be a choice between a red and blue shirt. Options are organized into OptionGroups. An OptionGroup is a class that knows the type of options that are being handled, in this case, color and the options available.

TODO: update how we deal with combinations of options across option groups and potential identifiers for each combination, for example size OptionGroup having Small and Large combined with a color OptionGroup having Red and Blue. The following shows the possible combinations. Of course adding more that two column increased the possible combinations.

color	size	available	indentifier(s)
blue	small	yes	SKU smbl
blue	large	no	N/A
red	small	yes	SKU smrd
red	large	yes	SKU lgrd

Identifiers

Identifiers are any type of system used to identify a product or variations of the product. Examples of identifiers are SKUs, ISBN, UPC, EAN or ASIN. It is possible for a single product to have more than one identifier assigned to it. For example, Test Driven Development by Kent Beck, has the ISBN-10 0321146530, ISBN-13 978-0321146533 and ASIN 0785342146530.

- Core
- Billing
- Cart
- Order
- Partner
- Pricing
- Product
- Store
- Taxonomy

Vespolina has a decoupled setup. This way you won't need to install a full webshop if you only need an inventory system for instance.

Read more about the components

CHAPTER 5

Framework integration

5.1 Symfony integration

5.1.1 Bundles

Store

The store bundle leverages most of the other Vespolina Bundles. It provides controllers, views and more.

Configuration

Read the more about the *Configuration* in the reference.

StoreBundle

Bla bla.

Read more about the *Store Bundle*

5.1.2 Configuration reference

TODO: Describe the configuration options for each bundle

StoreBundle

StoreBundle

TODO

Intro text

Read the *full reference*

5.1.3 VespolinaCommerceBundle

To make implementation easy, we choose to bundle to most used components in a single Symfony Bundle: CommerceBundle.

In this Bundle you'll find bindings for Cart, Checkout, Order, Process and Product.

Because of the decoupled setup of Vespolina, we are not bound to a single framework. At first we provide an integration for Symfony, but integration with other frameworks is possible.

Read about the Symfony integration

CHAPTER 6

About this documentation

This documentation is still a work in progress, but the goal is to follow the same rules and standards from Symfony development process.

- Github
- Website
- IRC Channel
- Dev mailing list

CHAPTER 7

Contribute

Want to contribute to this documentation? You can by submiting issues on github for typos, grammer mistakes, bugs in example code and so on.

CHAPTER 8

Indices and tables

- *genindex*
- *search*