SPIKE

Package’s research for shop building

SoftGroup Python Team 2

Introduction

Task: Build a shop, product – dogs.

Main requirements:

1. Start page should contain list of available products
2. Customer could view detailed info about each product, add it to shopping cart, delete it from shopping cart and finally place an order
3. Customer could't buy without registration
4. Admin pannel shoud contain options: add, delete, edit each product(content, add and delete images, rearrange their order)
5. Manager could track orders from admin pannel: view that they had been placed, detailed info '(customer's contacts, number of ordered items), edit status(processed, waiting for payment, closed etc.), add or remove items.
6. After creating an empty database there should be a script which automatically fill in database with products. (All breeds and their detailed description need to be parsed from wikipedia, images – downloaded using Google API)

Additional technical requirements:

1. White as few lines of code as possible
2. Use Django framework and packages for it

We've decided to seach via https://djangopackages.org/ for e-commerce package which would contain all (or almost all) required functionality and have opportunity to implement addition functionality if needed.

List of packages that we had been tried:

1. **DJANGO-OSCAR** - Domain-driven e-commerce for Django.
2. **DJANGO SHOP** - A Django based shop system.
3. **SATCHLESS** - E-commerce for Python.
4. **SATCHMO** – ecommerce framework built on the Django.
5. **CARTRIDGE** - Ecommerce for Mezzanine.

DJANGO-OSCAR

Oscar is an e-commerce framework for Django designed for building domain-driven sites. It is structured such that any part of the core functionality can be customized to suit the needs of project. One of the biggest advantages of Oscar is its extendible core. Any part of framework's functionality can be customized and extended, any class can be overridden or replaced. This feature allows Oscar to handle any business requirements and solve complex e-commerce issues quickly and efficiently.

Initially, our team would like to use this framework. Firstly, we went to the site and entered https://djangopackages.org to find E-commerce. Looking at the various Django packages selection was set for an Oscar.

This package was not a bad rating among other packages Django, it has 2119 stars. The package has a stable version. As well as recent findings in the project we are using Python 3.5.0, we drew attention to the fact that Oscar also supports Python 3.0. The important role played by the fact that Oscar has its own admin.

Also that we liked is that Oscar has a built-in application such as:

- Customer. Here the user can create, it has its own dashboard, which contains the purchase history. Also, the user can purchase goods without registration.

- Basket. The basket app handles shopping baskets, which essentially are a collection of products that hopefully end up being ordered.

- Dashboard. The dashboard is the backend interface for managing the store. That includes the product catalogue, orders and stock, offers etc. It is intended as a complete replacement of the Django admin interface.

- Payment. Often a shop will have a single mechanism for taking payment, such as integrating with a payment gateway or using PayPal. However more complicated projects will allow users to combine several different payment sources such as bankcards, business accounts and gift cards.

- Shipping. Oscar provides classes for free shipping, fixed charge shipping, pre-order and per-product item charges and weight-based charges. It is provides a mechanism for determining which shipping methods are available to the user.

- Order. Orders can be processed in many ways, including manual process and fully automated process. For instance, in manual process a worker in a warehouse may download a picking slip from the dashboard and mark products as shipped when they have been put in the van. Fully automated process, where files are transferred between the merchant and the fulfillment partner to indicate shipping statuses.

Advantages:

1. Development status – stable;

2. Supports Python3.0;
3. Popular on GitHub for developers;
4. Support Bootstrap templates;
5. There are its own applications;
6. You don’t need to develop your own e-commerce site. It's a lot of work to re-invent the wheel.

Disadvantages:

1. Too many unnecessary applications that have to be removed;
2. Documentation is good but not enough examples;
3. Oscar used for large projects, where you need to create a lot of goods;
4. Small community;
5. Customizing is very hard;
6. Unconvinient defauld interface(not as intuitive as expected)

We tried the sandbox(demo-shop) which was there and it was working - first thing we noticed was that it crashes if we try to search e.g. with word "count" it gives an server error. We got stuck in payment process with no way to get back.

Oscar is excellent framework for large projects, when it’s necessary to download a huge amount of goods. This framework can be used when you need to create high boot shop. Otherwise, it’s not much suitable for easy shopping and you have to spend a lot of time seaching wia documentaion to find an answer how to priperly customise it.

Conclusion:

Our team decided not use this framework. For our store we do not need so many functions, which are represented in Oscars.

DJANGO-SHOP

Official documentation of django-shop tells: this e-commerce package is good decision to start developing store. The framework is very well written, flexible, compact and uncomplicated. He supports python 3.5 and django version 1.9 and later. Django-shop well documented. He was integrated with django CMS, implemented with responsive images, bootstrap-3 grid system, panels, accordion and carousels.

Django-shop offers a set of basic functions:

1. Catalog to display a list of products and product presentation cards.
2. Some methods to add product to cart.
3. Functions remove items from the Recycle Bin or change their number.
4. A set of classes to change the value of the basket.
5. Collection of forms, where customers can add personal data, shipping and payment information
6. Converting basket in order.
7. Presentation of the list of orders in account.
8. The backend application is a tool to help monitor the status of orders.

All of these functions are necessary to build a real e-commerce site. However, the calculation of tax, for example, may vary in different ways between different legislations, and consequently, is not part of the framework. The same applies to the vouchers, discounts, delivery costs. These parts must be precisely defined sellers may be difficult to implement, so it is best to implement their individual modules.

Django-shop requires a description of your product instead of the predefined fields in the model. Products may differ from each other, and the simulation itself is not always easy. Some products can be sold in parts, others - set. Trying to define a set of fields in the Product model and its logic, which would be suitable for all online stores - it is impossible. Instead django-shop offers to describe the product with the required fields.

Django-shop has multilanguage. The products offered for different regions, usually require a description in their native language. For the same set of attributes of these products can be easily modeled using the fields that are to be translated. This allows to quite easily creating a multilingual internet-shop. Django-shop supports multiple currencies. He shipped with the currency type, using his own money arithmetic. This adds an additional layer of security, because no one can accidentally summarize the different currencies. These types of money always know how to present themselves in a variety of local conditions, bringing their number with the correct currency symbol. They also provide a number of special "no prices", which are haves as a "zero", it’s convenient for gratuitous objects.

Advantages:

1. Supports Python 3
2. Containes all modules that we need
3. Free license
4. Hopefull and promissing documentation

Disadvantages:

1. To set up a test default site you need to install Docker, and some other packages which was lond and irritative.
2. Many functions but hard to customise
3. Uses the built-in django admin interface. While this interface is not perfect, it is the one every django developers knows
4. Unsuitable documentation on GitHub

Conclusion:

Hard to customise, you need to spend much time to install and this is irritative and discourage.

SATCHLESS

Satchless is a low-level library that provides base classes, interfaces and patterns to work with while implementing a store. Its goal is to provide you with well-tested core logic to build business logic and UIs upon.

When our team began to search for alternatives to OSCAR, we noticed Satchless.

Advantages:

1. Small, compact
2. Supports Python3

Disadvantages:

1. Doesn’t a framework
2. Don’t contain at least half of functionality
3. Required writing to many own code
4. Bad documentation
5. Low activity on GitHub
6. Django 1.3 support
7. Still in early development

Conclusion:

To small for our purposes. Looks like a dead project

SATCHMO

Satchmo is an eCommerce framework created in Django which allows you to develop unique and robust online stores. One of the most well known eCommerce solution, that’s why we pay attention on it

Advantages:

1. Popular
2. Framework
3. Containes main functionality we need

   - As many images per product as you would like
   - Unlimited categories and sub categories
   - The customer model allows you to :view order history, update account profile online, reset user passwords

Disadvantages:

1. Hard installation (which is too long and irritative. You need to install for about 12 packages)
2. Doesn’t support Python 3
3. Official site doesn’t work, so you couldn’t search for information

Conclusion:

You’re in the code, and hours go by because the Product models.py has 1570 lines and it turns out it was a problem with **PriceAdjustmentCalc**, whatever that does.

We decided not to implement it because it’s really hard to find documentation so it would be hard to use it.

CARTRIDGE/MEZZANINE

Cartridge is a shopping cart application built using the Django framework. It is BSD licensed, and designed to provide a clean and simple base for developing e-commerce websites. It purposely does not include every conceivable feature of an e-commerce website; instead, Cartridge focuses on providing core features common to most e-commerce websites.

This specific focus stems from the idea that every e-commerce website is different, is tailored to the particular business and products at hand, and should therefore be as easy to customize as possible. Cartridge achieves this goal with a code-base that is as simple as possible and implements only the core features of an e-commerce website.

Cartridge extends the Mezzanine. Mezzanine is a powerful, consistent, and flexible content management platform. Built using the Django framework, Mezzanine provides a simple yet highly extensible architecture that encourages diving in and hacking on the code. Mezzanine is BSD licensed and supported by a diverse and active community.

In some ways, Mezzanine resembles tools such as Wordpress, providing an intuitive interface for managing pages, blog posts, form data, store products, and other types of content. But Mezzanine is also different. Unlike many other platforms that make extensive use of modules or reusable applications, Mezzanine provides most of its functionality by default. This approach yields a more integrated and efficient platform.

Advantages:

1. Easy to install and configure
2. Supports Python 3, package is regularly being updated.
3. Well documented. Documentaion highly organized and widespread.
4. Code clear, easy readable, has comments in hardly understandable places.
5. Contains all main functionality that we need.
6. Unnecessary modules could easy be removed.
7. Built-in test suite

Disadvantages:

1. Beta status (but after some search we discover that it has this status for more than 3 years but commits on GIT are regular)

Installation requires only two commannds and you could start working then, getting default package as a perfect beginning of your own project. The first impression after installation was very positive. Default interface for customer and admin pages is intuitively clear so that you could easy add and remove every package or feature you need.

We already have Bootstrap template and one of the main pluses was that they had been fastly implemented into this CMS.

Conclusion:

All these factors persuaded us to choose this platform as a perfect base for our future project(-s).