Building Ecommerce Projects with Moltin

- DREW MCLELLAN -

- PHPSW JULY 2015 -

Hello!

I’m Drew McLellan.

Lead dev on Perch CMS.

Publisher of 24ways.org

@drewm github.com/drewm

Open Source and crappy!

Hosted and restrictive!

Ecommerce sucks.

Bespoke and full of sadness :(

Enterprisey and useless!

Time spent on wrangling ecommerce is time stolen from finessing the UX.

Payment gateways are all different.

(AND MOST ARE TERRIBLE)

Compromise all the way down

To gain:	You lose:
Off-the-shelf functionality	UX flexibility
Ease of hosted SaaS	Seamless integration
Choice of gateway	Your hair
Web, app, PoS integration	Entire contents of your bank account

What could the answer possibly be?

Moltin, Inc moltin.com

LIVE CHAT HELP & SUPPORT DOCS

LOGIN REGISTER

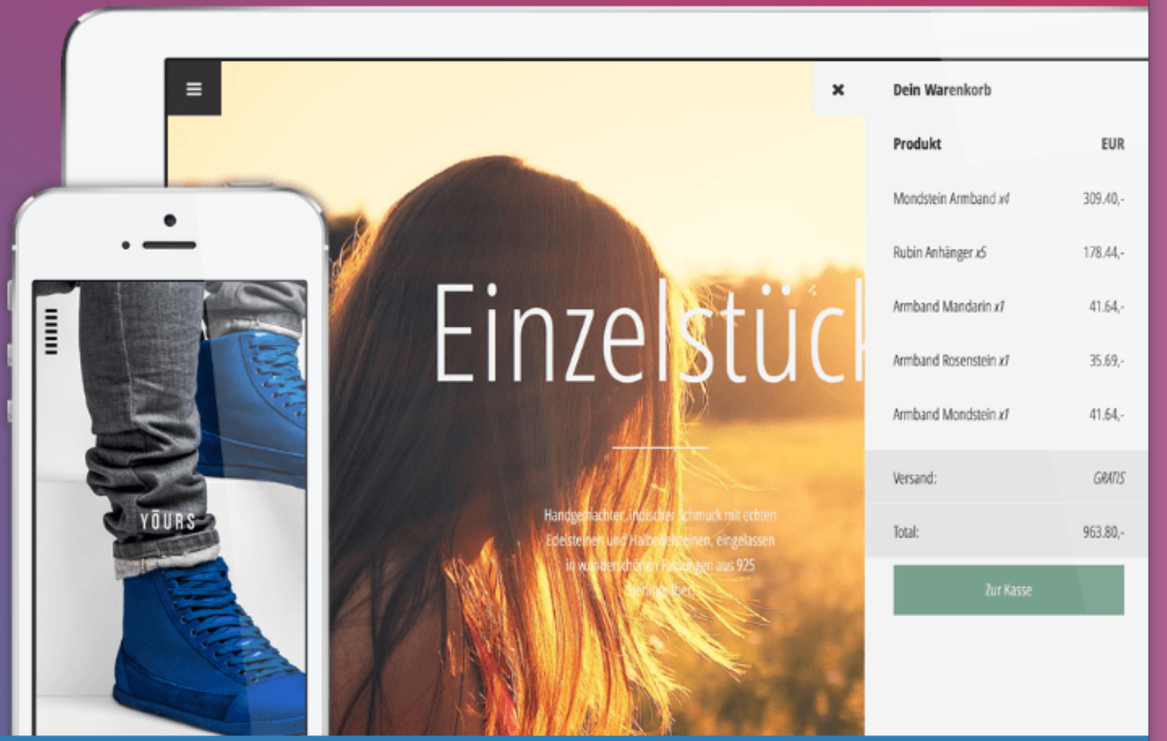
moltin

FEATURES HOW IT WORKS PRICING MORE

The quicker way to build eCommerce applications.

Inventory, cart, checkout, payments & more through a simple API, **moltin** was built from the ground up to power any website or mobile application in minutes.

GET STARTED



Join over 3,600 developers. It will only take a minute

GET STARTED

Moltin is an ecommerce web service API.

Products
Inventory
Carts
Checkout
Customers
Promotions
Shipping
Orders & Receipts
Currencies
Taxes

It works like this

1. Sign up at moltin.com
2. Create a store
3. Get an API key & secret
4. Install PHP SDK with Composer
5. Start building your store

Install the SDK

```
composer require moltin/php-sdk:dev-version1
```

Instantiate all the things

```php
use Moltin\SDK\Request\CURL as Request;
use Moltin\SDK\Storage\Session as Storage;

$moltin = new \Moltin\SDK\SDK(new Storage(), new Request());
```

Authenticate

```
$result = \Moltin::Authenticate('ClientCredentials', [
        'client_id'     => '<YOUR CLIENT ID>',
        'client_secret' => '<YOUR CLIENT SECRET>',
]);
```

Get stuff done

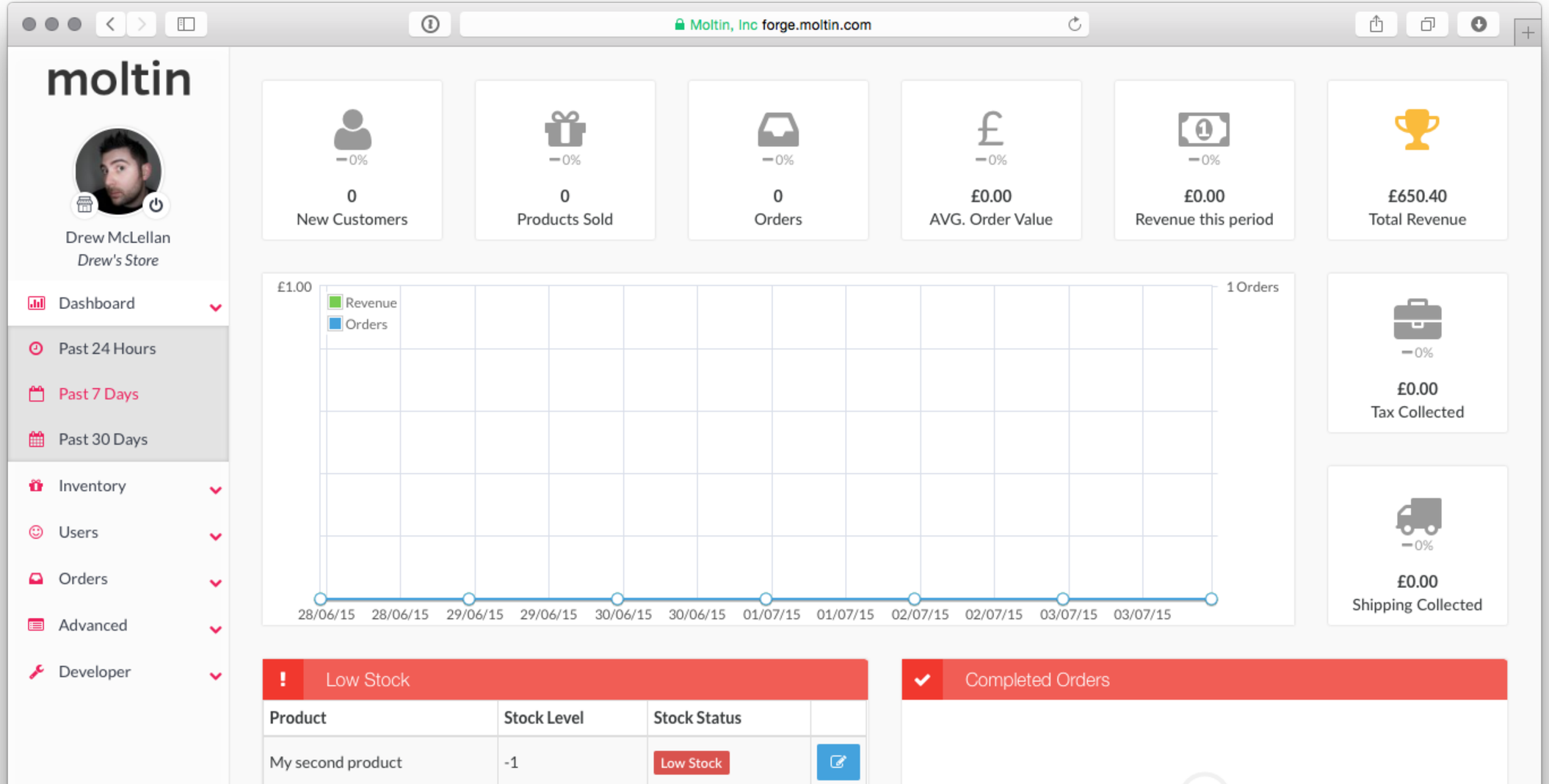
```php
// Create a product
$result = \Product::Create([…]);

// Update a product
$result = \Product::Update('<ID>', ['title' => 'Updated!')];

// Get a product
$result = Product::Get('<ID>');

// Delete a product
$result = Product::Delete('<ID>');
```

Forge Dashboard



Checkout

50+ Payment Providers

authorize.Net

DataCash

GoCardless

Mollie

NetBanx

PayPal

SagePay

SecPay / PayPoint.net

Stripe

WorldPay

Checkout

Calculates shipping, taxes, discounts.

Adjusts stock levels.

Generates orders and invoices.

Handles gateway API interactions*

* sort of.

Gateways

You don’t get total isolation from gateway weirdness.

You can’t *just swap* one gateway for another.

Still needs planning for payment flows to call the right Moltin API methods at the right times.

Moltin bills on API requests.

Billed on API requests

Free up to 30,000 per month.

300,000 reqs is $49

600,000 reqs is $99

...

5,000,000 reqs $499

So.

Advantages

You get almost complete control over the UX.

You don’t need to build all the boring stuff.

Multiple apps (website, mobile, PoS, etc) can all use the same backend with no extra work.

Swap out your front end easily.

Start adding inventory via Forge before site is built.

Disadvantages

Creates a very large dependancy compared to something self-hosted.

That comes with risk of downtime or service discontinuation.

Introduces network latency into many operations.

Not as flexible as something completely bespoke.

Questions?