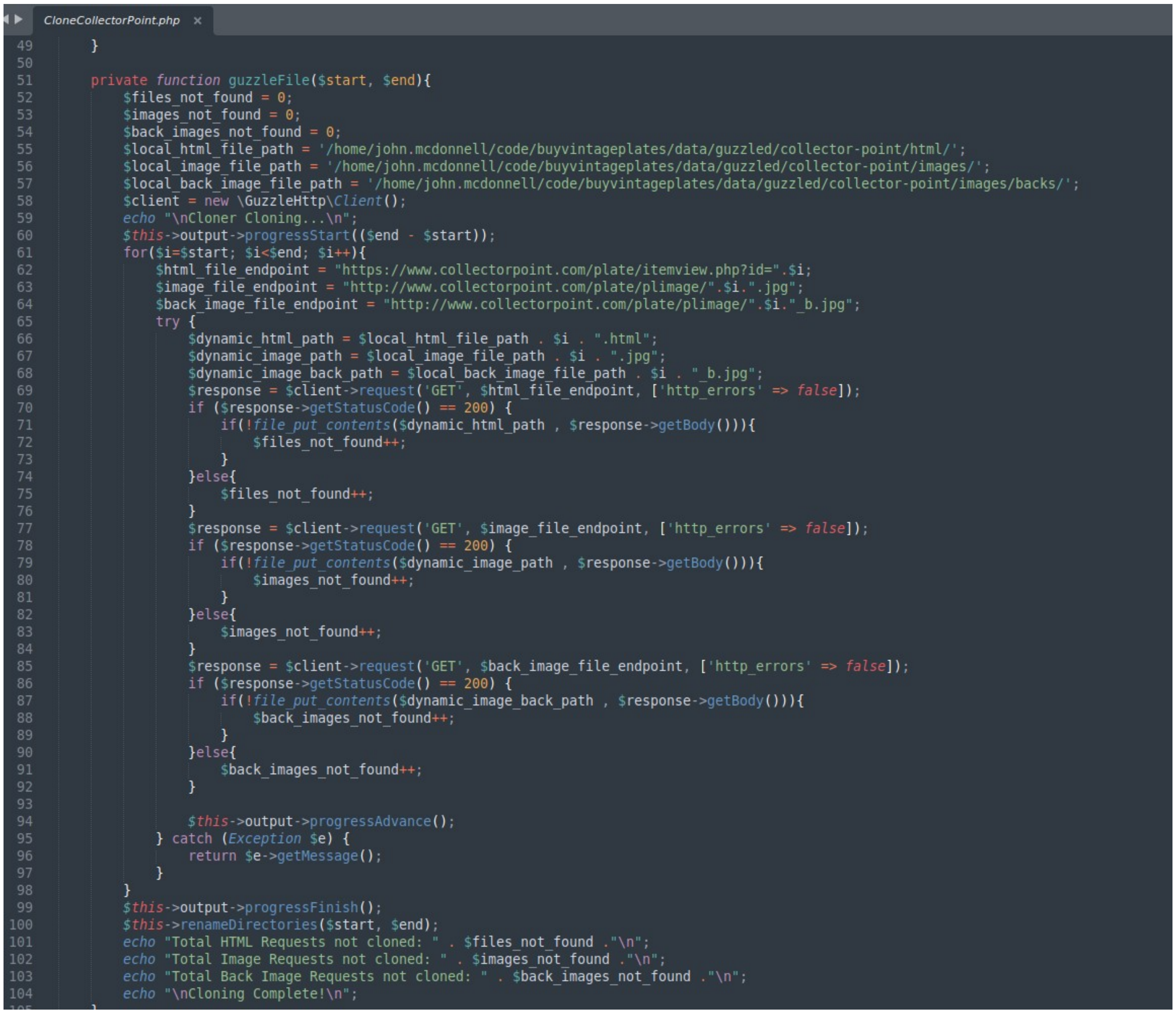
Problem: Create a solution to re-use existing data saving myself significant time listing products for sale on ebay, facebook marketplace, amazon, and mercari. Also use the system to track sales and various other metrics. Currently the system has two complete datasets of reference products that I often re-sell… Collectible Plates and Books. I will cover how I utilized laravel/vue/mysql/php to implement this system.

Step one: Acquire offline data for product references for Plates. In order to do this I built a custom set of laravel artisan commands that utilize curl through the guzzle library to grab each and every plate info page which is about 30,000 plates.

<u>Harvesting Command</u>

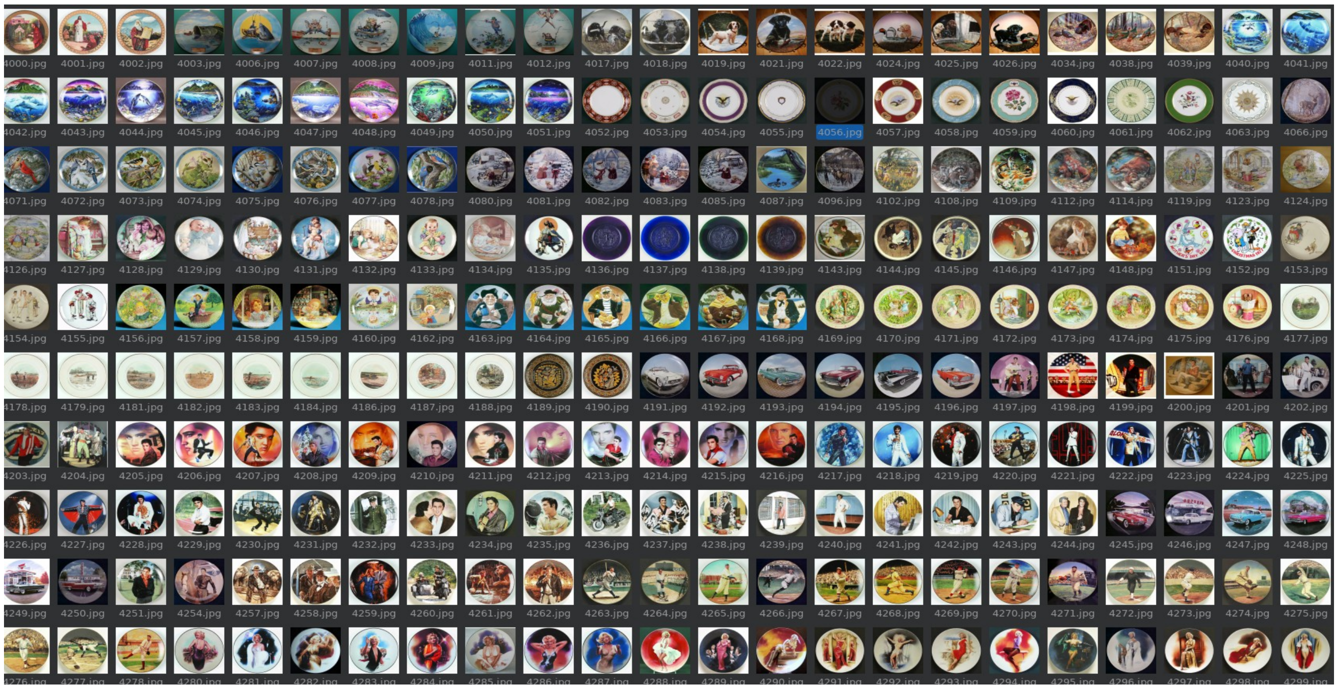


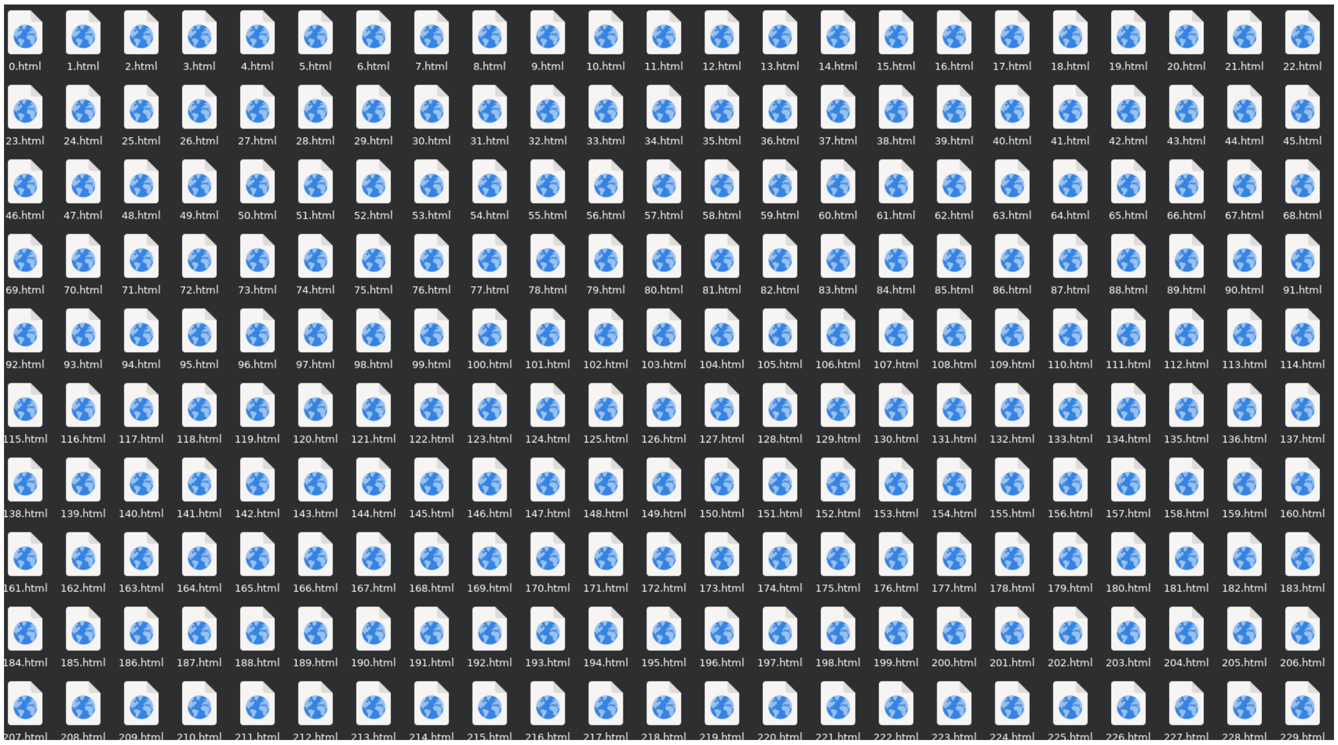
```php
    }

    private function guzzleFile($start, $end){
        $files_not_found = 0;
        $images_not_found = 0;
        $back_images_not_found = 0;
        $local_html_file_path = '/home/john.mcdonnell/code/buyvintageplates/data/guzzled/collector-point/html/';
        $local_image_file_path = '/home/john.mcdonnell/code/buyvintageplates/data/guzzled/collector-point/images/';
        $local_back_image_file_path = '/home/john.mcdonnell/code/buyvintageplates/data/guzzled/collector-point/images/backs/';
        $client = new \GuzzleHttp\Client();
        echo "\nCloner Cloning...\n";
        $this->output->progressStart(($end - $start));
        for($i=$start; $i<$end; $i++){
            $html_file_endpoint = "https://www.collectorpoint.com/plate/itemview.php?id=".$i;
            $image_file_endpoint = "http://www.collectorpoint.com/plate/plimage/".$i.".jpg";
            $back_image_file_endpoint = "http://www.collectorpoint.com/plate/plimage/".$i."_b.jpg";
            try {
                $dynamic_html_path = $local_html_file_path . $i . ".html";
                $dynamic_image_path = $local_image_file_path . $i . ".jpg";
                $dynamic_image_back_path = $local_back_image_file_path . $i . "_b.jpg";
                $response = $client->request('GET', $html_file_endpoint, ['http_errors' => false]);
                if ($response->getStatusCode() == 200) {
                    if(!file_put_contents($dynamic_html_path , $response->getBody())){
                        $files_not_found++;
                    }
                }else{
                    $files_not_found++;
                }
                $response = $client->request('GET', $image_file_endpoint, ['http_errors' => false]);
                if ($response->getStatusCode() == 200) {
                    if(!file_put_contents($dynamic_image_path , $response->getBody())){
                        $images_not_found++;
                    }
                }else{
                    $images_not_found++;
                }
                $response = $client->request('GET', $back_image_file_endpoint, ['http_errors' => false]);
                if ($response->getStatusCode() == 200) {
                    if(!file_put_contents($dynamic_image_back_path , $response->getBody())){
                        $back_images_not_found++;
                    }
                }else{
                    $back_images_not_found++;
                }

                $this->output->progressAdvance();
            } catch (Exception $e) {
                return $e->getMessage();
            }
        }
        $this->output->progressFinish();
        $this->renameDirectories($start, $end);
        echo "Total HTML Requests not cloned: " . $files_not_found ."\n";
        echo "Total Image Requests not cloned: " . $images_not_found ."\n";
        echo "Total Back Image Requests not cloned: " . $back_images_not_found ."\n";
        echo "\nCloning Complete!\n";
```

This command basically gives a progress bar and utilizes an improvised semi-restful design based off and existing sites URL structure to harvest sequential pages based on IDs. It then stores the raw HTML (contains all the plate data) inside the appropriately labeled data directory. The command was structured to run small sets or large sets of harvesting passes through $start-$end being the ID range… Obviously I didn’t want to invoke an accidental DOS so I did small passes at harvesting.

Harvesting Command Results

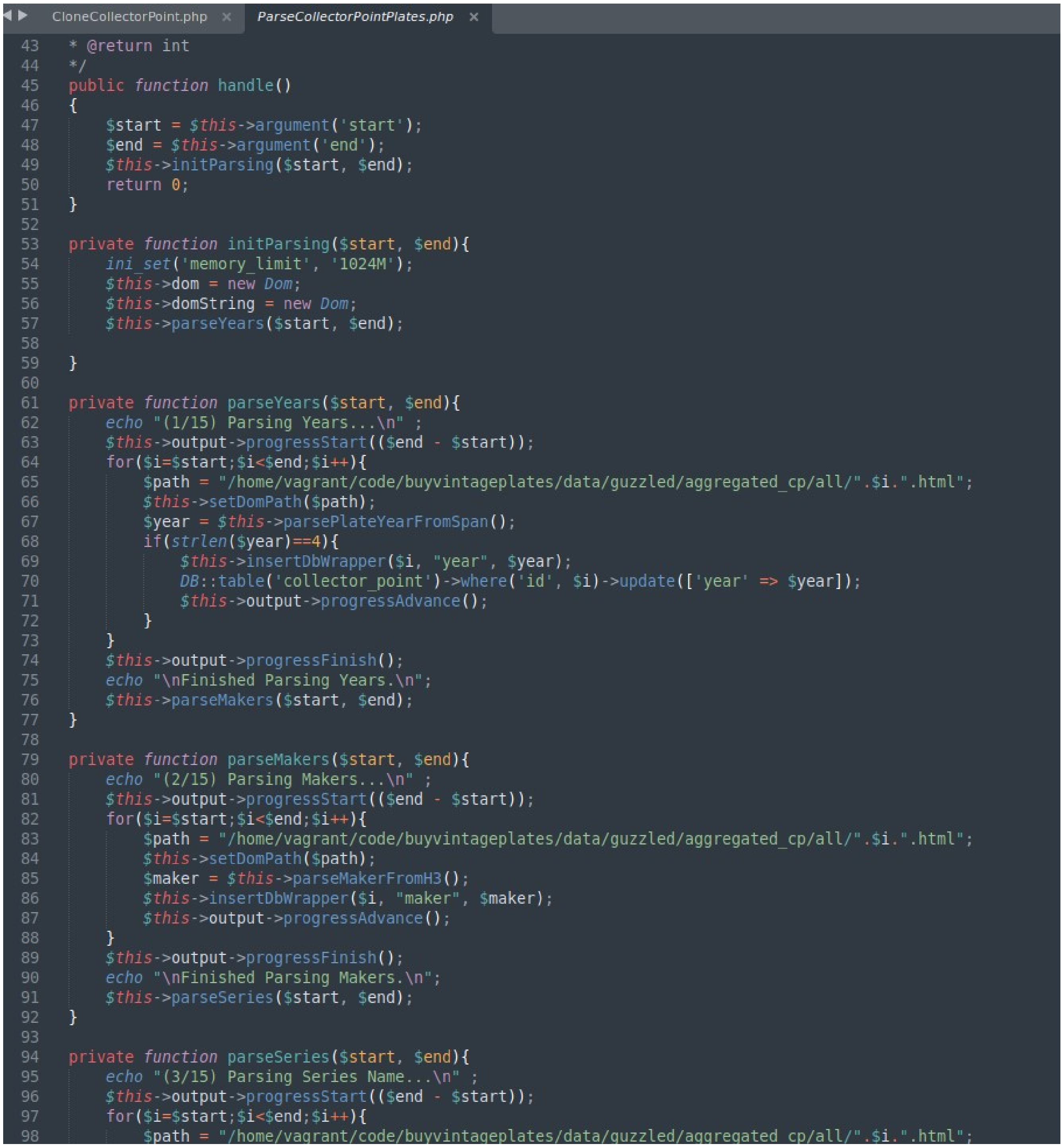
The results of the command are pictured here:


4000.jpg 4001.jpg 4002.jpg 4003.jpg 4006.jpg 4007.jpg 4008.jpg 4009.jpg 4011.jpg 4012.jpg 4017.jpg 4018.jpg 4019.jpg 4021.jpg 4022.jpg 4024.jpg 4025.jpg 4026.jpg 4034.jpg 4038.jpg 4039.jpg 4040.jpg 4041.jpg
4042.jpg 4043.jpg 4044.jpg 4045.jpg 4046.jpg 4047.jpg 4048.jpg 4049.jpg 4050.jpg 4051.jpg 4052.jpg 4053.jpg 4054.jpg 4055.jpg 4056.jpg 4057.jpg 4058.jpg 4059.jpg 4060.jpg 4061.jpg 4062.jpg 4063.jpg 4066.jpg
4071.jpg 4072.jpg 4073.jpg 4074.jpg 4075.jpg 4076.jpg 4077.jpg 4078.jpg 4080.jpg 4081.jpg 4082.jpg 4083.jpg 4085.jpg 4087.jpg 4096.jpg 4102.jpg 4108.jpg 4109.jpg 4112.jpg 4114.jpg 4119.jpg 4123.jpg 4124.jpg
4126.jpg 4127.jpg 4128.jpg 4129.jpg 4130.jpg 4131.jpg 4132.jpg 4133.jpg 4134.jpg 4135.jpg 4136.jpg 4137.jpg 4138.jpg 4139.jpg 4143.jpg 4144.jpg 4145.jpg 4146.jpg 4147.jpg 4148.jpg 4151.jpg 4152.jpg 4153.jpg
4154.jpg 4155.jpg 4156.jpg 4157.jpg 4158.jpg 4159.jpg 4160.jpg 4162.jpg 4163.jpg 4164.jpg 4165.jpg 4166.jpg 4167.jpg 4168.jpg 4169.jpg 4170.jpg 4171.jpg 4172.jpg 4173.jpg 4174.jpg 4175.jpg 4176.jpg 4177.jpg
4178.jpg 4179.jpg 4181.jpg 4182.jpg 4183.jpg 4184.jpg 4186.jpg 4187.jpg 4188.jpg 4189.jpg 4190.jpg 4191.jpg 4192.jpg 4193.jpg 4194.jpg 4195.jpg 4196.jpg 4197.jpg 4198.jpg 4199.jpg 4200.jpg 4201.jpg 4202.jpg
4203.jpg 4204.jpg 4205.jpg 4206.jpg 4207.jpg 4208.jpg 4209.jpg 4210.jpg 4211.jpg 4212.jpg 4213.jpg 4214.jpg 4215.jpg 4216.jpg 4217.jpg 4218.jpg 4219.jpg 4220.jpg 4221.jpg 4222.jpg 4223.jpg 4224.jpg 4225.jpg
4226.jpg 4227.jpg 4228.jpg 4229.jpg 4230.jpg 4231.jpg 4232.jpg 4233.jpg 4234.jpg 4235.jpg 4236.jpg 4237.jpg 4238.jpg 4239.jpg 4240.jpg 4241.jpg 4242.jpg 4243.jpg 4244.jpg 4245.jpg 4246.jpg 4247.jpg 4248.jpg
4249.jpg 4250.jpg 4251.jpg 4254.jpg 4257.jpg 4258.jpg 4259.jpg 4260.jpg 4261.jpg 4262.jpg 4263.jpg 4264.jpg 4265.jpg 4266.jpg 4267.jpg 4268.jpg 4269.jpg 4270.jpg 4271.jpg 4272.jpg 4273.jpg 4274.jpg 4275.jpg


0.html 1.html 2.html 3.html 4.html 5.html 6.html 7.html 8.html 9.html 10.html 11.html 12.html 13.html 14.html 15.html 16.html 17.html 18.html 19.html 20.html 21.html 22.html
23.html 24.html 25.html 26.html 27.html 28.html 29.html 30.html 31.html 32.html 33.html 34.html 35.html 36.html 37.html 38.html 39.html 40.html 41.html 42.html 43.html 44.html 45.html
46.html 47.html 48.html 49.html 50.html 51.html 52.html 53.html 54.html 55.html 56.html 57.html 58.html 59.html 60.html 61.html 62.html 63.html 64.html 65.html 66.html 67.html 68.html
69.html 70.html 71.html 72.html 73.html 74.html 75.html 76.html 77.html 78.html 79.html 80.html 81.html 82.html 83.html 84.html 85.html 86.html 87.html 88.html 89.html 90.html 91.html
92.html 93.html 94.html 95.html 96.html 97.html 98.html 99.html 100.html 101.html 102.html 103.html 104.html 105.html 106.html 107.html 108.html 109.html 110.html 111.html 112.html 113.html 114.html
115.html 116.html 117.html 118.html 119.html 120.html 121.html 122.html 123.html 124.html 125.html 126.html 127.html 128.html 129.html 130.html 131.html 132.html 133.html 134.html 135.html 136.html 137.html
138.html 139.html 140.html 141.html 142.html 143.html 144.html 145.html 146.html 147.html 148.html 149.html 150.html 151.html 152.html 153.html 154.html 155.html 156.html 157.html 158.html 159.html 160.html
161.html 162.html 163.html 164.html 165.html 166.html 167.html 168.html 169.html 170.html 171.html 172.html 173.html 174.html 175.html 176.html 177.html 178.html 179.html 180.html 181.html 182.html 183.html
184.html 185.html 186.html 187.html 188.html 189.html 190.html 191.html 192.html 193.html 194.html 195.html 196.html 197.html 198.html 199.html 200.html 201.html 202.html 203.html 204.html 205.html 206.html

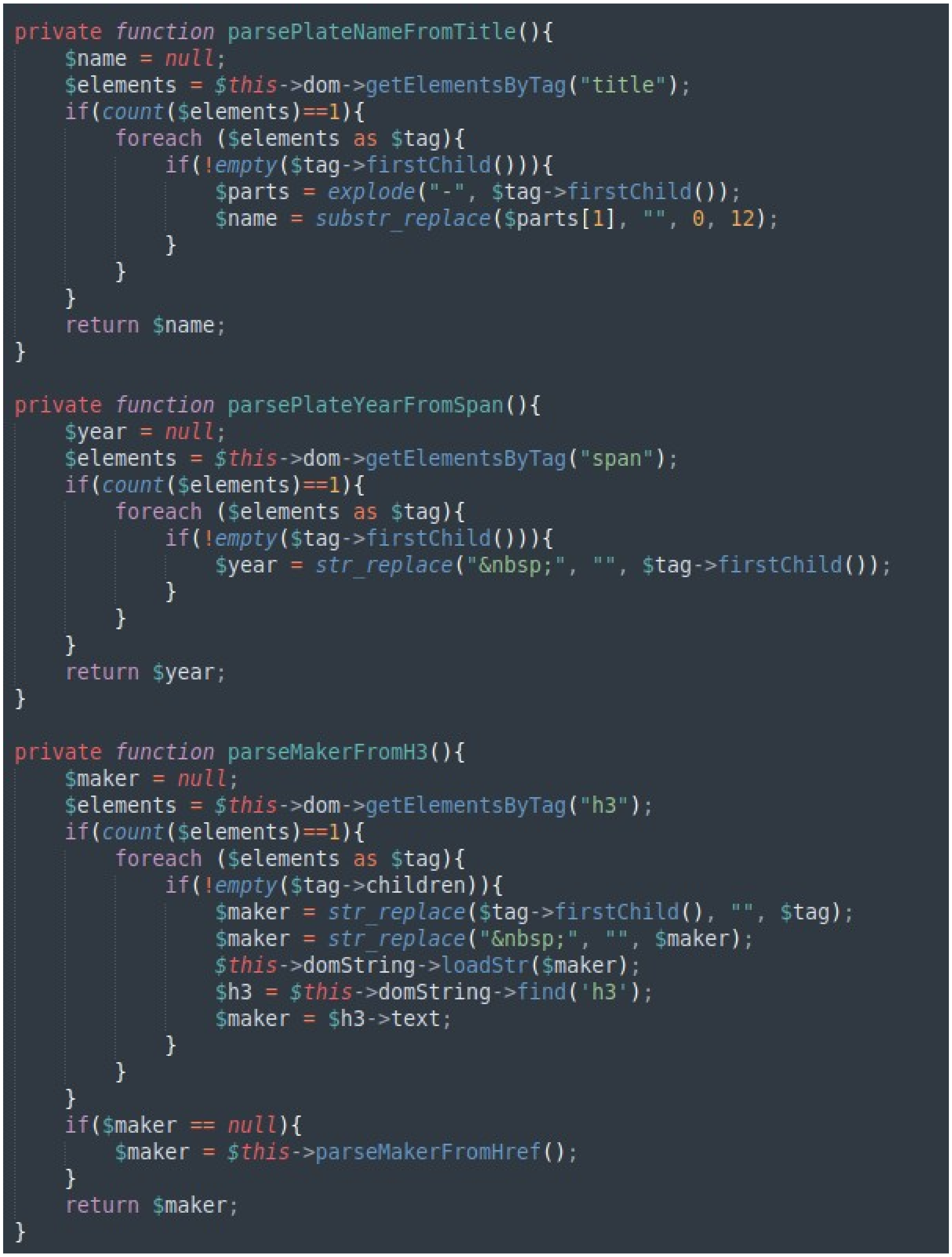
Step two: After acquiring all the data the next step was to take the plate data from the raw HTML and store it into a database table. To do this I created a another command that implemented a basic HTML DOM parser to parse out the various desired data fields according to certain custom parsing rules that had to be created to target specific parts of the data. This often required very specifically and meticulously crafted rule-sets to isolate the data against many varying conditions.

<u>The Parsing Command</u>



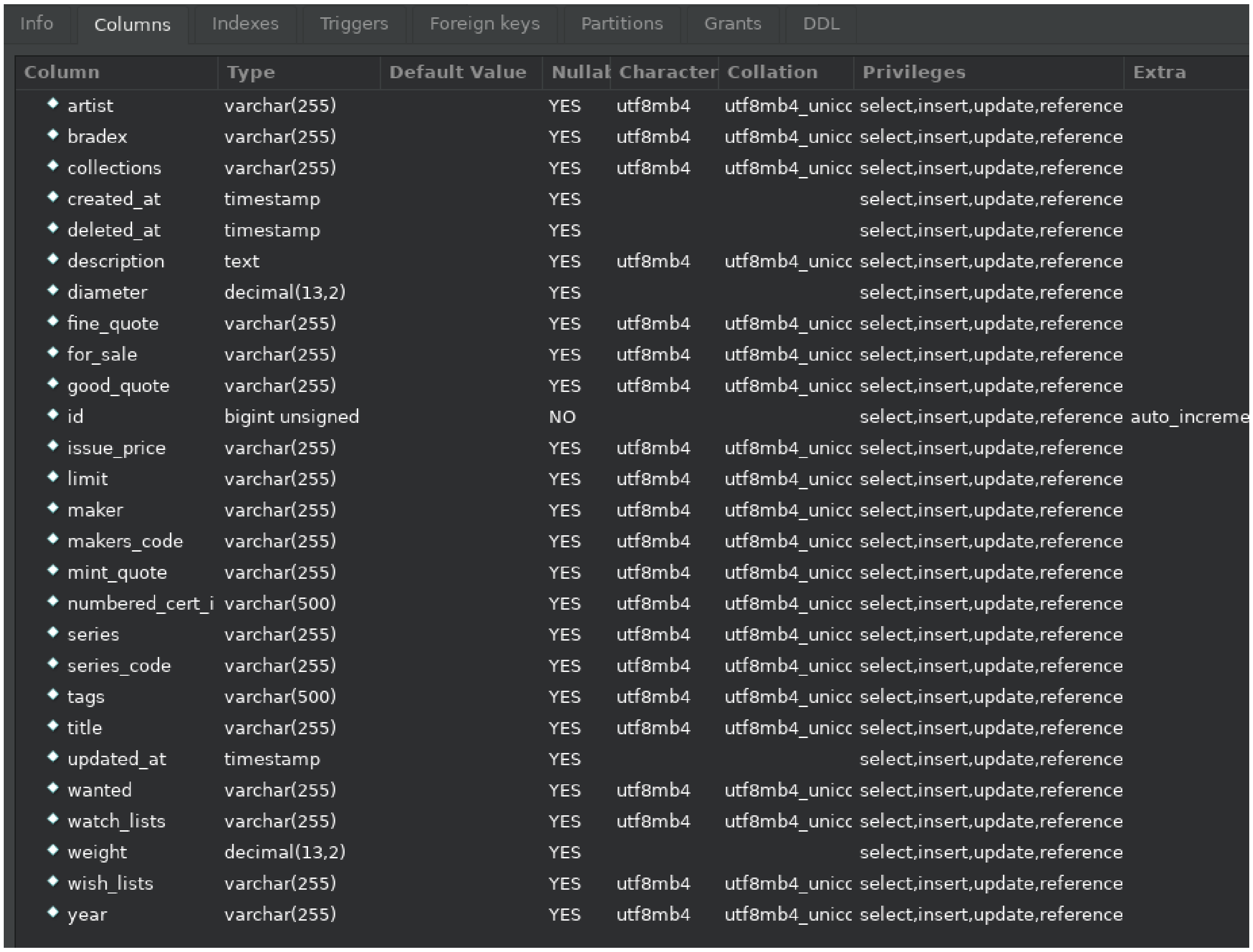
```php
 * @return int
 */
public function handle()
{
    $start = $this->argument('start');
    $end = $this->argument('end');
    $this->initParsing($start, $end);
    return 0;
}

private function initParsing($start, $end){
    ini_set('memory_limit', '1024M');
    $this->dom = new Dom;
    $this->domString = new Dom;
    $this->parseYears($start, $end);

}

private function parseYears($start, $end){
    echo "(1/15) Parsing Years...\n" ;
    $this->output->progressStart(($end - $start));
    for($i=$start;$i<$end;$i++){
        $path = "/home/vagrant/code/buyvintageplates/data/guzzled/aggregated_cp/all/".$i.".html";
        $this->setDomPath($path);
        $year = $this->parsePlateYearFromSpan();
        if(strlen($year)==4){
            $this->insertDbWrapper($i, "year", $year);
            DB::table('collector_point')->where('id', $i)->update(['year' => $year]);
            $this->output->progressAdvance();
        }
    }
    $this->output->progressFinish();
    echo "\nFinished Parsing Years.\n";
    $this->parseMakers($start, $end);
}

private function parseMakers($start, $end){
    echo "(2/15) Parsing Makers...\n" ;
    $this->output->progressStart(($end - $start));
    for($i=$start;$i<$end;$i++){
        $path = "/home/vagrant/code/buyvintageplates/data/guzzled/aggregated_cp/all/".$i.".html";
        $this->setDomPath($path);
        $maker = $this->parseMakerFromH3();
        $this->insertDbWrapper($i, "maker", $maker);
        $this->output->progressAdvance();
    }
    $this->output->progressFinish();
    echo "\nFinished Parsing Makers.\n";
    $this->parseSeries($start, $end);
}

private function parseSeries($start, $end){
    echo "(3/15) Parsing Series Name...\n" ;
    $this->output->progressStart(($end - $start));
    for($i=$start;$i<$end;$i++){
        $path = "/home/vagrant/code/buyvintageplates/data/guzzled/aggregated_cp/all/".$i.".html";
```

The Parsing Command rule-set



```php
private function parsePlateNameFromTitle(){
    $name = null;
    $elements = $this->dom->getElementsByTag("title");
    if(count($elements)==1){
        foreach ($elements as $tag){
            if(!empty($tag->firstChild())){
                $parts = explode("-", $tag->firstChild());
                $name = substr_replace($parts[1], "", 0, 12);
            }
        }
    }
    return $name;
}

private function parsePlateYearFromSpan(){
    $year = null;
    $elements = $this->dom->getElementsByTag("span");
    if(count($elements)==1){
        foreach ($elements as $tag){
            if(!empty($tag->firstChild())){
                $year = str_replace("&nbsp;", "", $tag->firstChild());
            }
        }
    }
    return $year;
}

private function parseMakerFromH3(){
    $maker = null;
    $elements = $this->dom->getElementsByTag("h3");
    if(count($elements)==1){
        foreach ($elements as $tag){
            if(!empty($tag->children)){
                $maker = str_replace($tag->firstChild(), "", $tag);
                $maker = str_replace("&nbsp;", "", $maker);
                $this->domString->loadStr($maker);
                $h3 = $this->domString->find('h3');
                $maker = $h3->text;
            }
        }
    }
    if($maker == null){
        $maker = $this->parseMakerFromHref();
    }
    return $maker;
}
```

The parsing command could run all the methods from the init or individual methods as they were tested and developed to improve accuracy. The command parsed the isolated data to a corresponding mysql table… i.e. the parsed year → year column – parsed plate → plate name etc. This created a pretty rudimentary and basic table show below with an approximate ***<u>non-normalized size of 6.5 mb</u>***



Column	Type	Default Value	Nullak	Character	Collation	Privileges	Extra
artist	varchar(255)		YES	utf8mb4	utf8mb4_unicc	select,insert,update,reference	
bradex	varchar(255)		YES	utf8mb4	utf8mb4_unicc	select,insert,update,reference	
collections	varchar(255)		YES	utf8mb4	utf8mb4_unicc	select,insert,update,reference	
created_at	timestamp		YES			select,insert,update,reference	
deleted_at	timestamp		YES			select,insert,update,reference	
description	text		YES	utf8mb4	utf8mb4_unicc	select,insert,update,reference	
diameter	decimal(13,2)		YES			select,insert,update,reference	
fine_quote	varchar(255)		YES	utf8mb4	utf8mb4_unicc	select,insert,update,reference	
for_sale	varchar(255)		YES	utf8mb4	utf8mb4_unicc	select,insert,update,reference	
good_quote	varchar(255)		YES	utf8mb4	utf8mb4_unicc	select,insert,update,reference	
id	bigint unsigned		NO			select,insert,update,reference	auto_increme
issue_price	varchar(255)		YES	utf8mb4	utf8mb4_unicc	select,insert,update,reference	
limit	varchar(255)		YES	utf8mb4	utf8mb4_unicc	select,insert,update,reference	
maker	varchar(255)		YES	utf8mb4	utf8mb4_unicc	select,insert,update,reference	
makers_code	varchar(255)		YES	utf8mb4	utf8mb4_unicc	select,insert,update,reference	
mint_quote	varchar(255)		YES	utf8mb4	utf8mb4_unicc	select,insert,update,reference	
numbered_cert_i	varchar(500)		YES	utf8mb4	utf8mb4_unicc	select,insert,update,reference	
series	varchar(255)		YES	utf8mb4	utf8mb4_unicc	select,insert,update,reference	
series_code	varchar(255)		YES	utf8mb4	utf8mb4_unicc	select,insert,update,reference	
tags	varchar(500)		YES	utf8mb4	utf8mb4_unicc	select,insert,update,reference	
title	varchar(255)		YES	utf8mb4	utf8mb4_unicc	select,insert,update,reference	
updated_at	timestamp		YES			select,insert,update,reference	
wanted	varchar(255)		YES	utf8mb4	utf8mb4_unicc	select,insert,update,reference	
watch_lists	varchar(255)		YES	utf8mb4	utf8mb4_unicc	select,insert,update,reference	
weight	decimal(13,2)		YES			select,insert,update,reference	
wish_lists	varchar(255)		YES	utf8mb4	utf8mb4_unicc	select,insert,update,reference	
year	varchar(255)		YES	utf8mb4	utf8mb4_unicc	select,insert,update,reference	

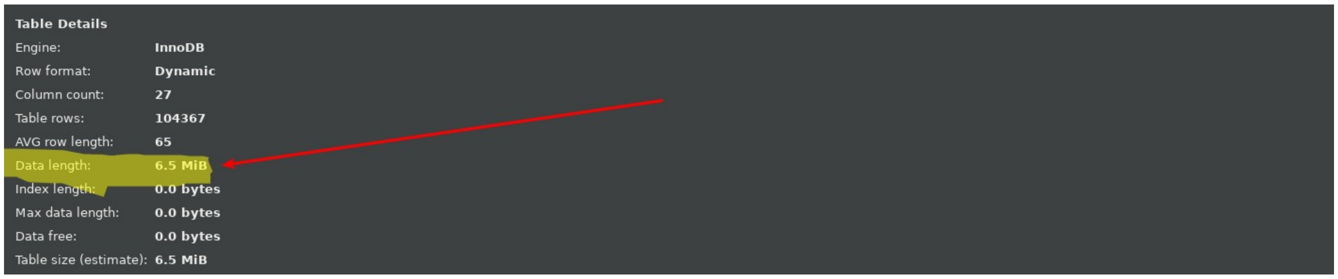
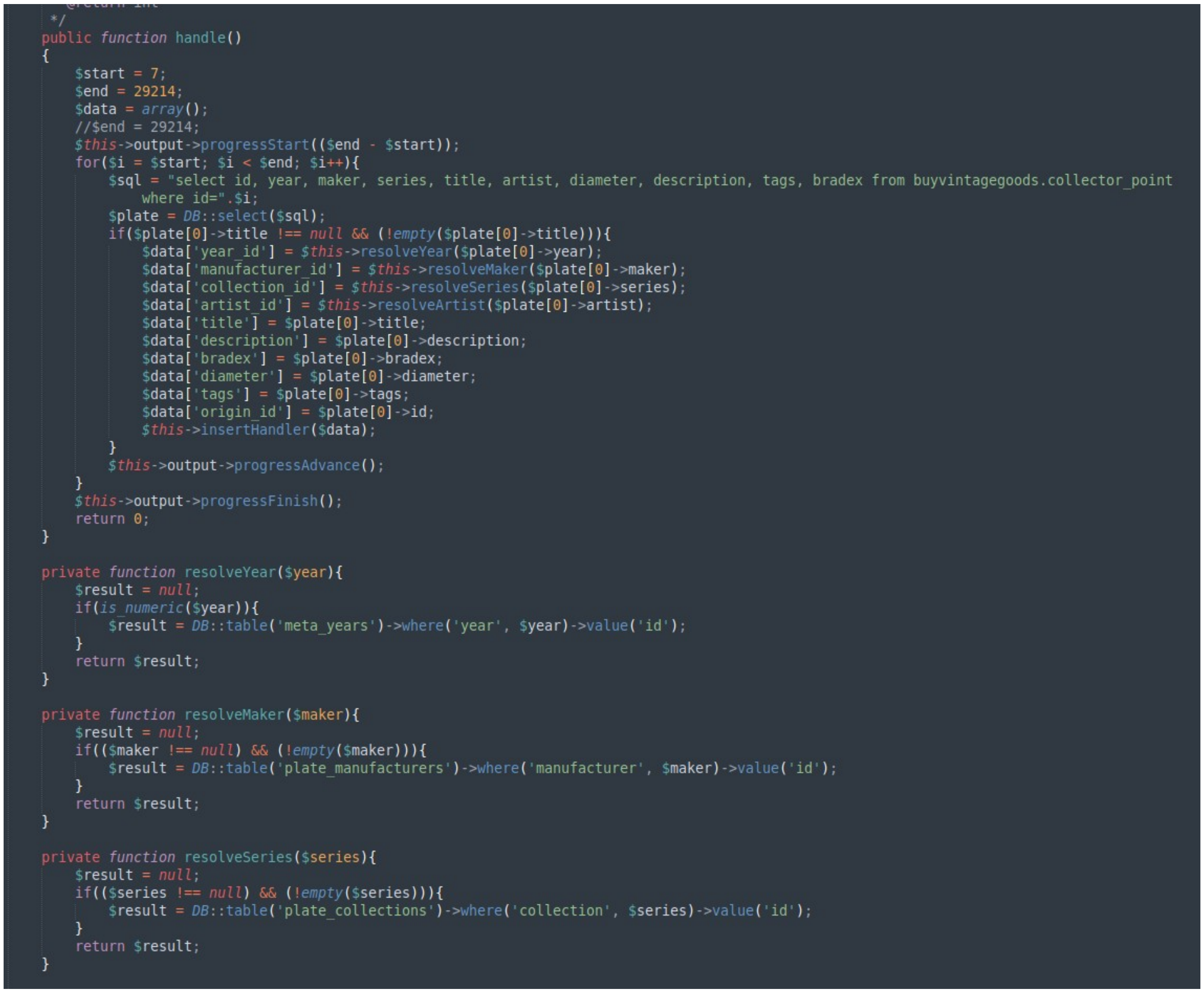


Table Details

Engine:	InnoDB
Row format:	Dynamic
Column count:	27
Table rows:	104367
AVG row length:	65
Data length:	6.5 MiB
Index length:	0.0 bytes
Max data length:	0.0 bytes
Data free:	0.0 bytes
Table size (estimate):	6.5 MiB

Step Three: Normalize the data into a newer more efficient table with relationships to other tables… For instance the Plate Artists and Manufacturers were moved to their own tables and their total size is less than 100 KB… I actually created a normalization command to resolve fields from one table to the other…



```
     */
    public function handle()
    {
        $start = 7;
        $end = 29214;
        $data = array();
        //$end = 29214;
        $this->output->progressStart(($end - $start));
        for($i = $start; $i < $end; $i++){
            $sql = "select id, year, maker, series, title, artist, diameter, description, tags, bradex from buyvintagegoods.collector_point
                where id=".$i;
            $plate = DB::select($sql);
            if($plate[0]->title !== null && (!empty($plate[0]->title))){
                $data['year_id'] = $this->resolveYear($plate[0]->year);
                $data['manufacturer_id'] = $this->resolveMaker($plate[0]->maker);
                $data['collection_id'] = $this->resolveSeries($plate[0]->series);
                $data['artist_id'] = $this->resolveArtist($plate[0]->artist);
                $data['title'] = $plate[0]->title;
                $data['description'] = $plate[0]->description;
                $data['bradex'] = $plate[0]->bradex;
                $data['diameter'] = $plate[0]->diameter;
                $data['tags'] = $plate[0]->tags;
                $data['origin_id'] = $plate[0]->id;
                $this->insertHandler($data);
            }
            $this->output->progressAdvance();
        }
        $this->output->progressFinish();
        return 0;
    }

    private function resolveYear($year){
        $result = null;
        if(is_numeric($year)){
            $result = DB::table('meta_years')->where('year', $year)->value('id');
        }
        return $result;
    }

    private function resolveMaker($maker){
        $result = null;
        if(($maker !== null) && (!empty($maker))){
            $result = DB::table('plate_manufacturers')->where('manufacturer', $maker)->value('id');
        }
        return $result;
    }

    private function resolveSeries($series){
        $result = null;
        if(($series !== null) && (!empty($series))){
            $result = DB::table('plate_collections')->where('collection', $series)->value('id');
        }
        return $result;
    }
```

Which resulted in the following normalized table(s)

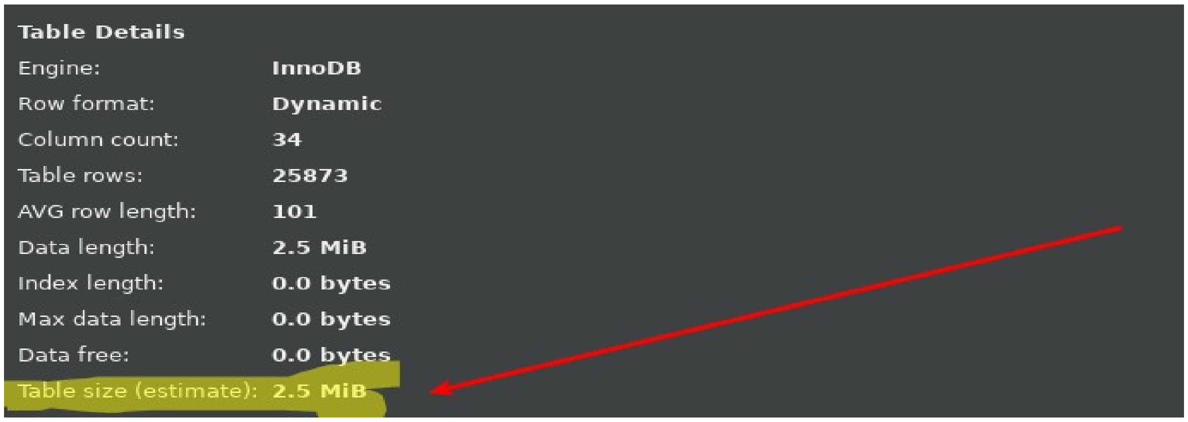
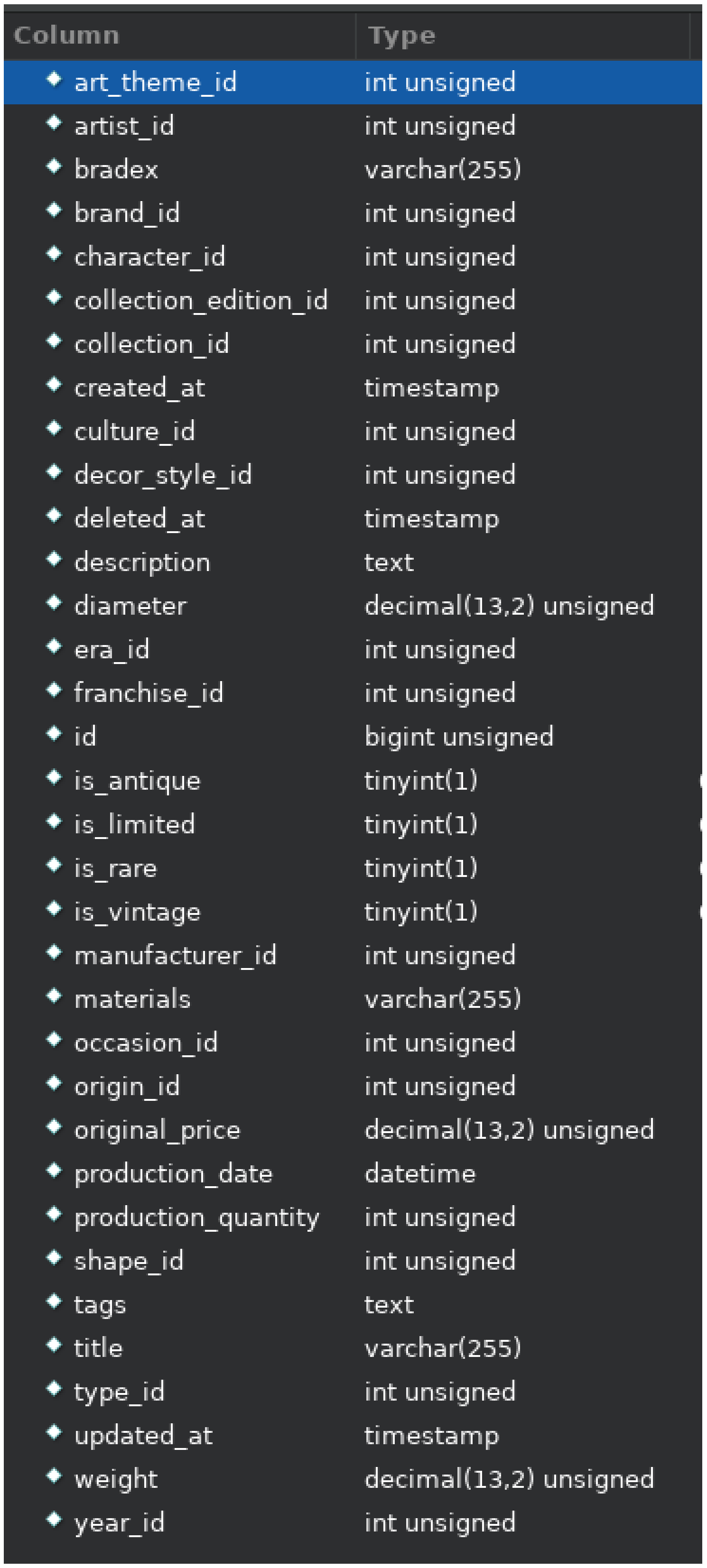
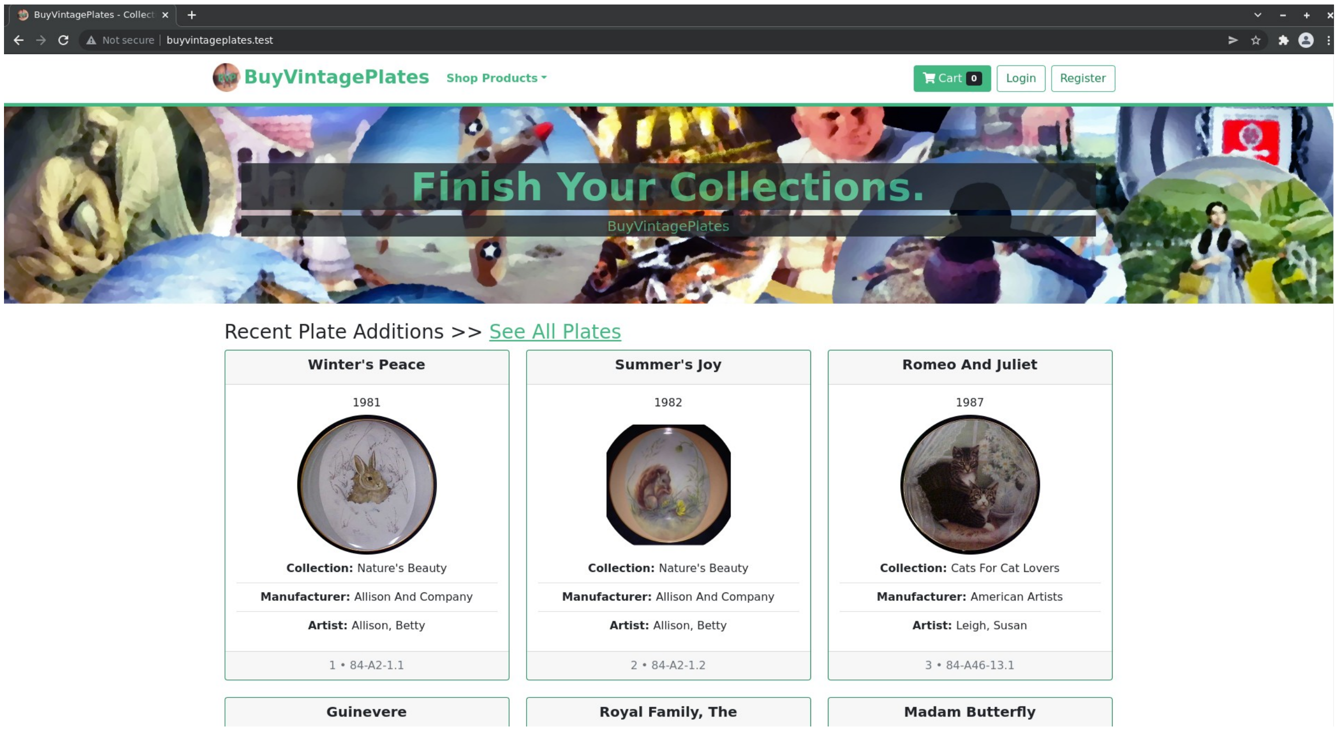


Table Details	
Engine:	InnoDB
Row format:	Dynamic
Column count:	34
Table rows:	25873
AVG row length:	101
Data length:	2.5 MiB
Index length:	0.0 bytes
Max data length:	0.0 bytes
Data free:	0.0 bytes
Table size (estimate):	2.5 MiB



Column	Type
art_theme_id	int unsigned
artist_id	int unsigned
bradex	varchar(255)
brand_id	int unsigned
character_id	int unsigned
collection_edition_id	int unsigned
collection_id	int unsigned
created_at	timestamp
culture_id	int unsigned
decor_style_id	int unsigned
deleted_at	timestamp
description	text
diameter	decimal(13,2) unsigned
era_id	int unsigned
franchise_id	int unsigned
id	bigint unsigned
is_antique	tinyint(1)
is_limited	tinyint(1)
is_rare	tinyint(1)
is_vintage	tinyint(1)
manufacturer_id	int unsigned
materials	varchar(255)
occasion_id	int unsigned
origin_id	int unsigned
original_price	decimal(13,2) unsigned
production_date	datetime
production_quantity	int unsigned
shape_id	int unsigned
tags	text
title	varchar(255)
type_id	int unsigned
updated_at	timestamp
weight	decimal(13,2) unsigned
year_id	int unsigned

Final Result



BuyVintagePlates - Collect

Not secure | buyvintageplates.test

BuyVintagePlates Shop Products

Cart 0 Login Register

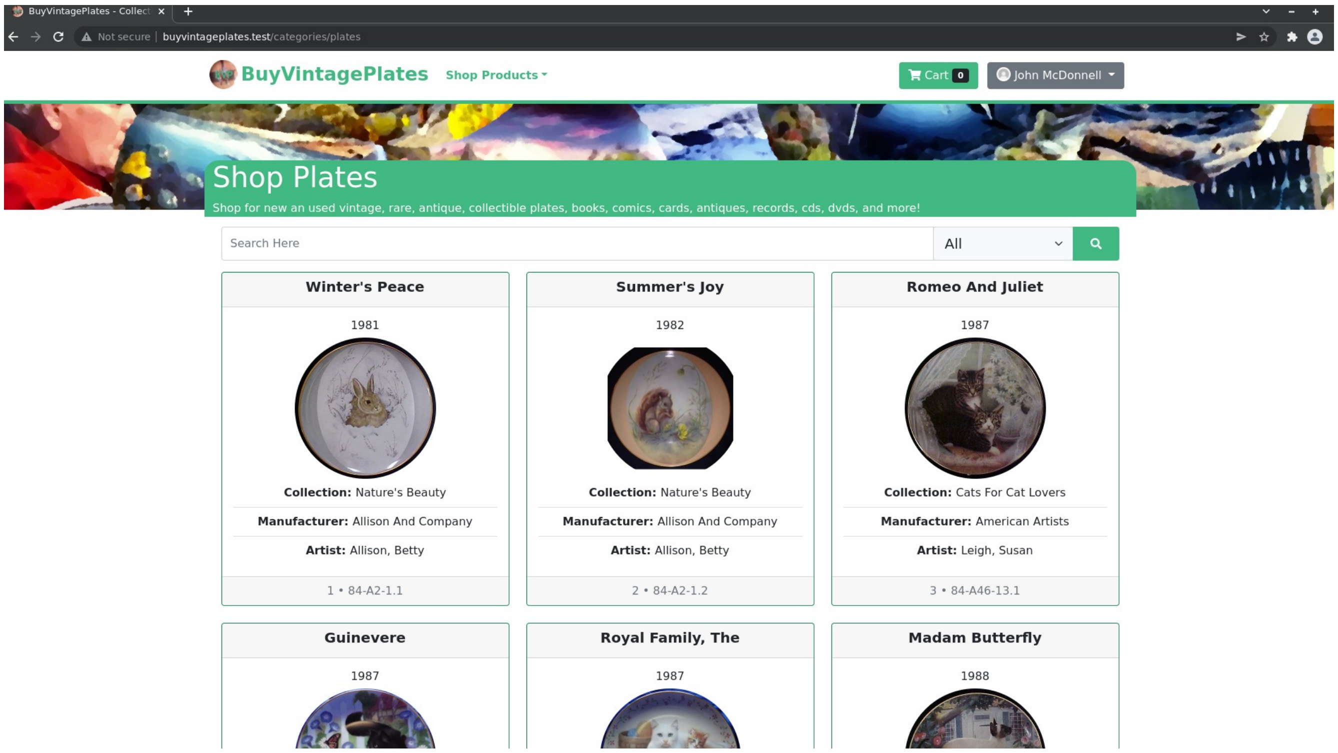
Finish Your Collections.

BuyVintagePlates

Recent Plate Additions >> See All Plates

Winter's Peace	Summer's Joy	Romeo And Juliet
1981	1982	1987
Collection: Nature's Beauty	**Collection:** Nature's Beauty	**Collection:** Cats For Cat Lovers
Manufacturer: Allison And Company	**Manufacturer:** Allison And Company	**Manufacturer:** American Artists
Artist: Allison, Betty	**Artist:** Allison, Betty	**Artist:** Leigh, Susan
1 • 84-A2-1.1	2 • 84-A2-1.2	3 • 84-A46-13.1

Guinevere | Royal Family, The | Madam Butterfly



BuyVintagePlates - Collect

Not secure | buyvintageplates.test/categories/plates

BuyVintagePlates Shop Products

Cart 0 John McDonnell

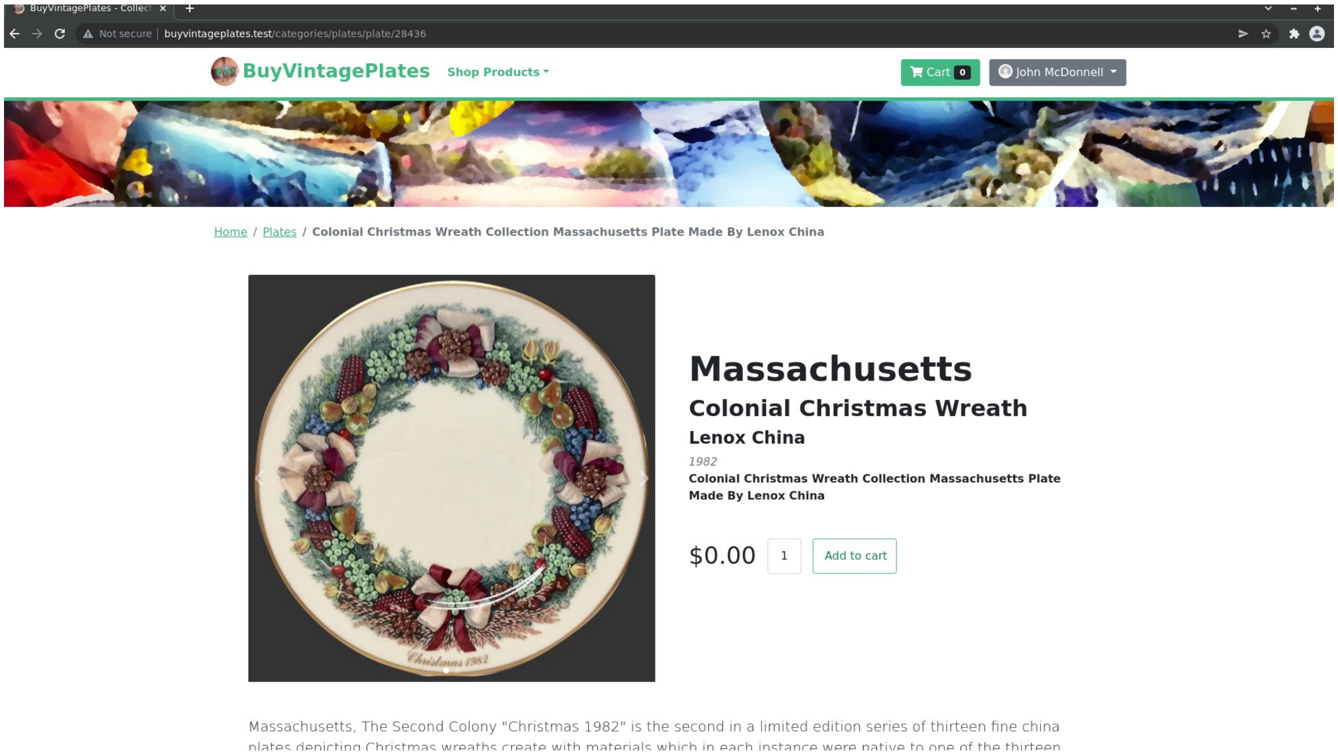
Shop Plates

Shop for new an used vintage, rare, antique, collectible plates, books, comics, cards, antiques, records, cds, dvds, and more!

Search Here | All

Winter's Peace	Summer's Joy	Romeo And Juliet
1981	1982	1987
Collection: Nature's Beauty	**Collection:** Nature's Beauty	**Collection:** Cats For Cat Lovers
Manufacturer: Allison And Company	**Manufacturer:** Allison And Company	**Manufacturer:** American Artists
Artist: Allison, Betty	**Artist:** Allison, Betty	**Artist:** Leigh, Susan
1 • 84-A2-1.1	2 • 84-A2-1.2	3 • 84-A46-13.1

Guinevere	Royal Family, The	Madam Butterfly
1987	1987	1988


BuyVintagePlates - Collect
Not secure | buyvintageplates.test/categories/plates/plate/28436
BuyVintagePlates
Shop Products
Cart 0
John McDonnell

Home / Plates / Colonial Christmas Wreath Collection Massachusetts Plate Made By Lenox China

Massachusetts

Colonial Christmas Wreath

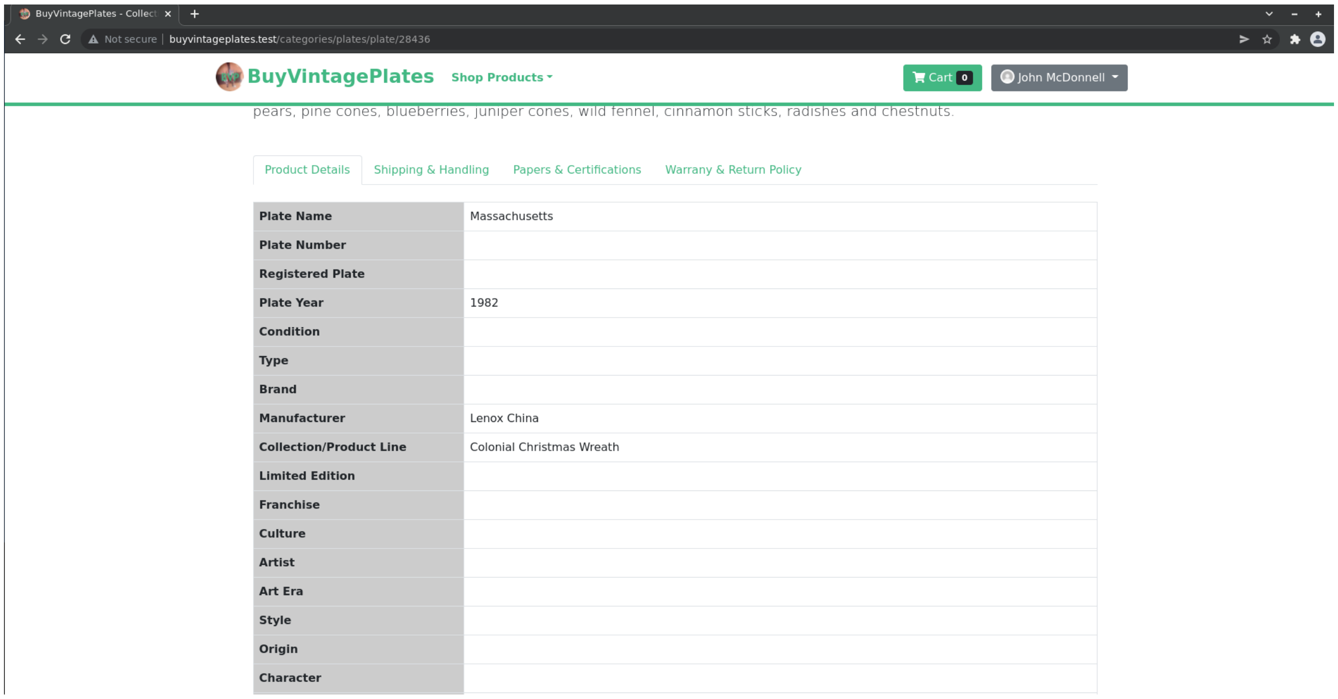
Lenox China

1982

Colonial Christmas Wreath Collection Massachusetts Plate Made By Lenox China

$0.00 1 Add to cart

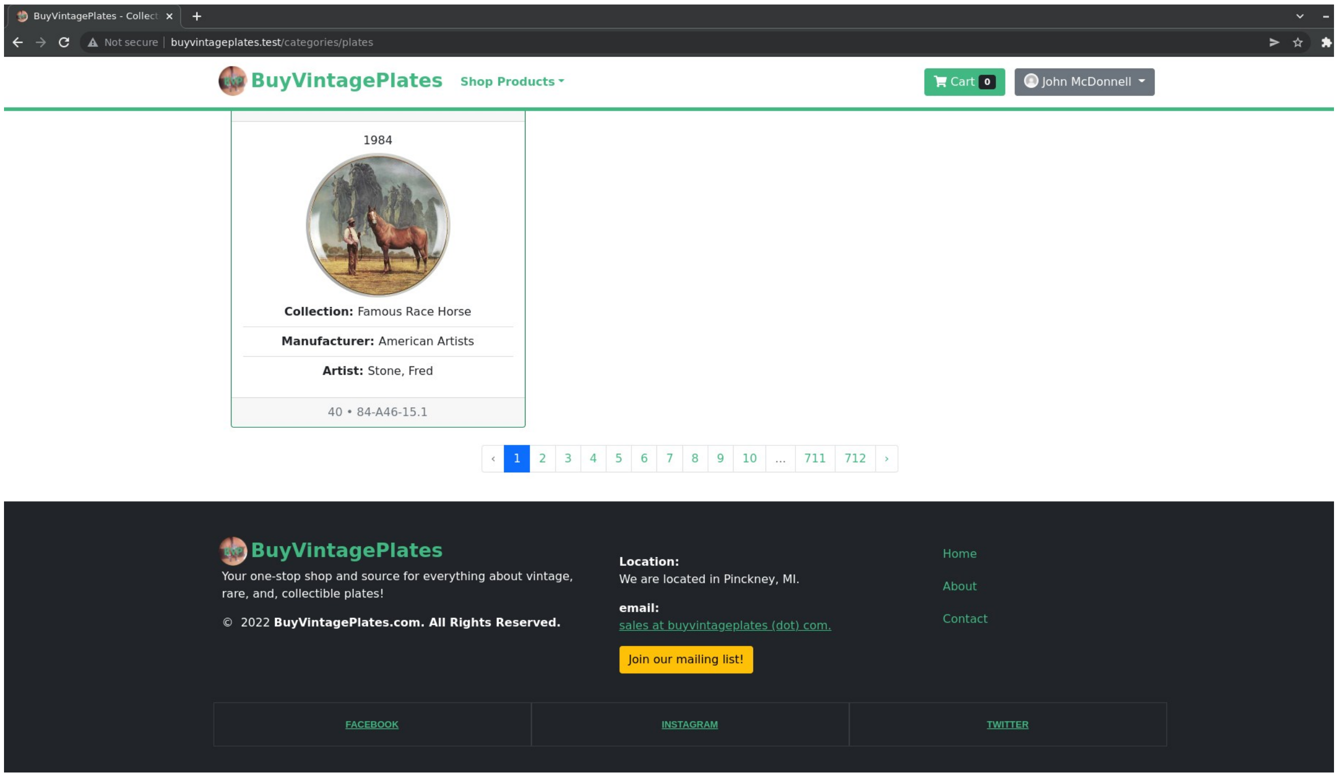
Massachusetts, The Second Colony "Christmas 1982" is the second in a limited edition series of thirteen fine china plates depicting Christmas wreaths create with materials which in each instance were native to one of the thirteen


BuyVintagePlates - Collect
Not secure | buyvintageplates.test/categories/plates/plate/28436
BuyVintagePlates
Shop Products
Cart 0
John McDonnell

pears, pine cones, blueberries, juniper cones, wild fennel, cinnamon sticks, radishes and chestnuts.

Product Details | Shipping & Handling | Papers & Certifications | Warranty & Return Policy

Plate Name	Massachusetts
Plate Number	
Registered Plate	
Plate Year	1982
Condition	
Type	
Brand	
Manufacturer	Lenox China
Collection/Product Line	Colonial Christmas Wreath
Limited Edition	
Franchise	
Culture	
Artist	
Art Era	
Style	
Origin	
Character	


BuyVintagePlates - Collect

Not secure | buyvintageplates.test/categories/plates

BuyVintagePlates Shop Products

Cart 0 | John McDonnell

1984

Collection: Famous Race Horse

Manufacturer: American Artists

Artist: Stone, Fred

40 • 84-A46-15.1

‹ 1 2 3 4 5 6 7 8 9 10 … 711 712 ›

BuyVintagePlates

Your one-stop shop and source for everything about vintage, rare, and, collectible plates!

© 2022 BuyVintagePlates.com. All Rights Reserved.

Location:
We are located in Pinckney, MI.

email:
sales at buyvintageplates (dot) com.

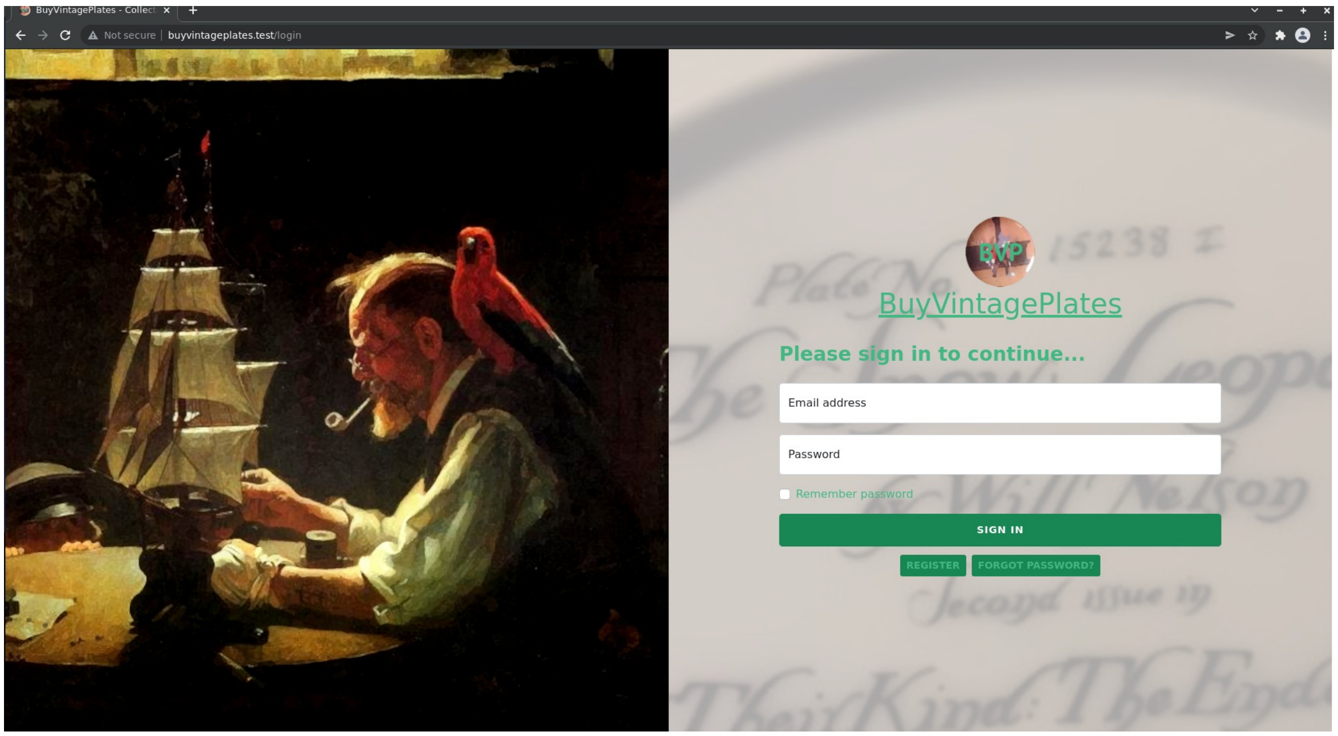
Join our mailing list!

Home

About

Contact

FACEBOOK | INSTAGRAM | TWITTER


BuyVintagePlates - Collect

Not secure | buyvintageplates.test/login

BuyVintagePlates

Please sign in to continue...

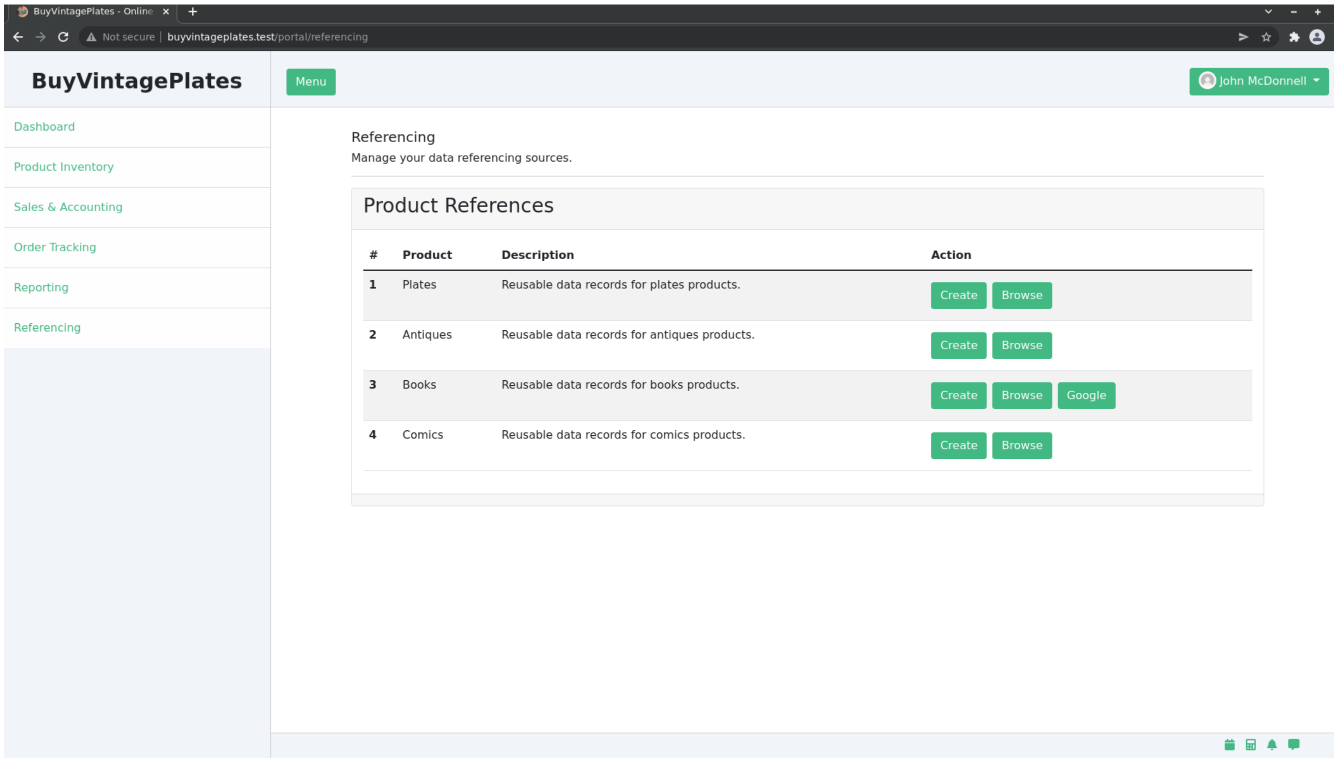
Email address

Password

Remember password

SIGN IN

REGISTER | FORGOT PASSWORD?



BuyVintagePlates - Online

Not secure | buyvintageplates.test/portal/referencing

BuyVintagePlates

Menu

John McDonnell

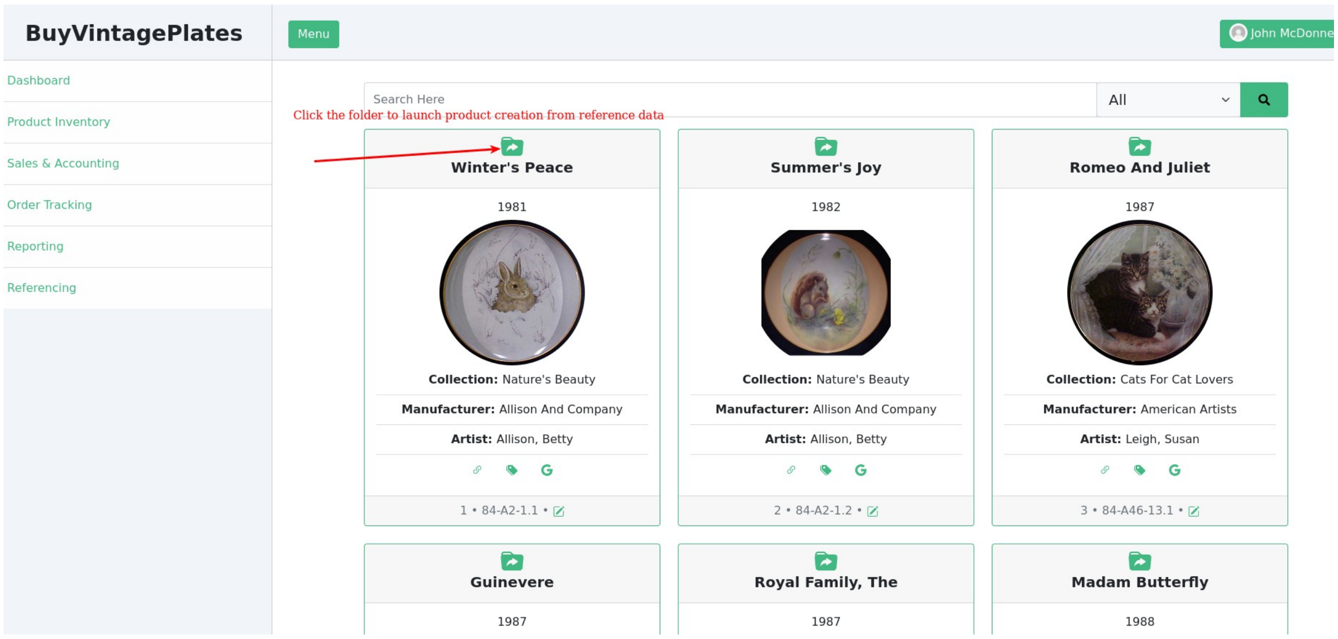
- Dashboard
- Product Inventory
- Sales & Accounting
- Order Tracking
- Reporting
- Referencing

Referencing

Manage your data referencing sources.

Product References

#	Product	Description	Action
1	Plates	Reusable data records for plates products.	Create Browse
2	Antiques	Reusable data records for antiques products.	Create Browse
3	Books	Reusable data records for books products.	Create Browse Google
4	Comics	Reusable data records for comics products.	Create Browse



BuyVintagePlates

Menu

John McDonne

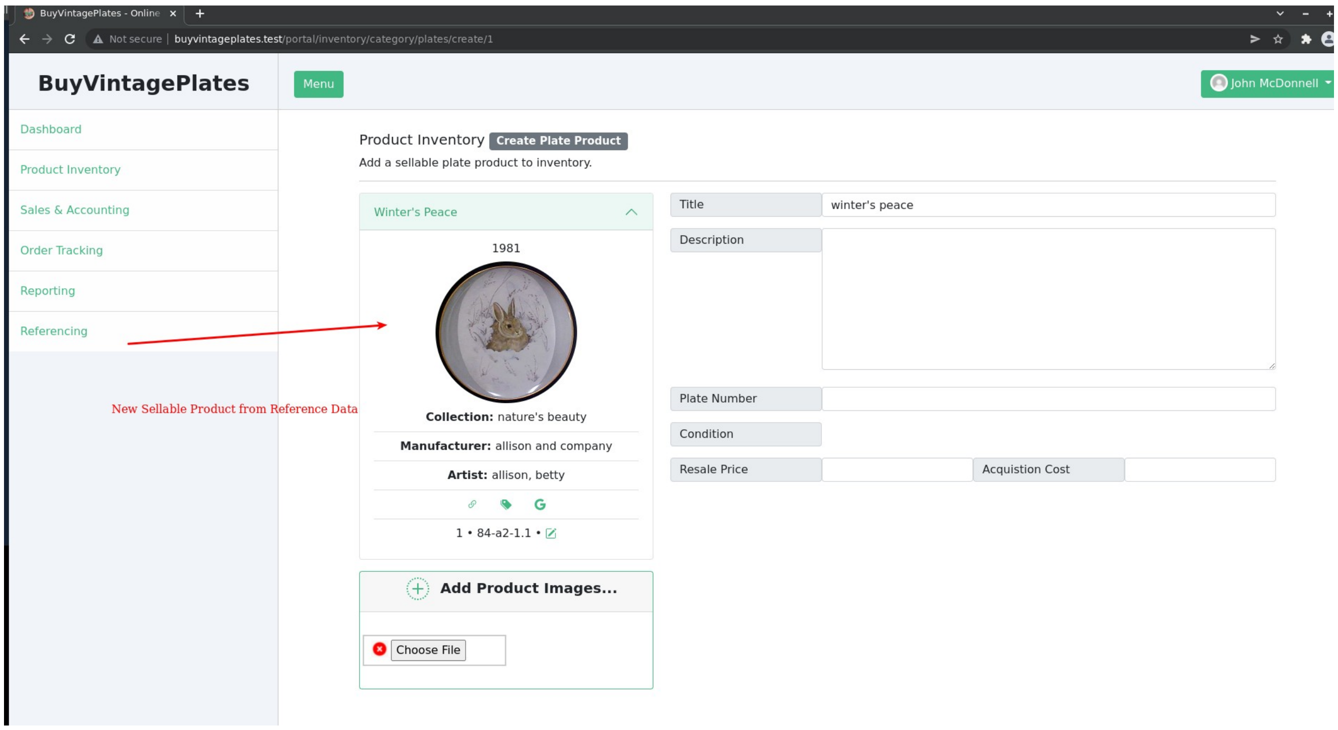
- Dashboard
- Product Inventory
- Sales & Accounting
- Order Tracking
- Reporting
- Referencing

Search Here | All

Click the folder to launch product creation from reference data

Winter's Peace	Summer's Joy	Romeo And Juliet
1981	1982	1987
Collection: Nature's Beauty	**Collection:** Nature's Beauty	**Collection:** Cats For Cat Lovers
Manufacturer: Allison And Company	**Manufacturer:** Allison And Company	**Manufacturer:** American Artists
Artist: Allison, Betty	**Artist:** Allison, Betty	**Artist:** Leigh, Susan
1 • 84-A2-1.1 •	2 • 84-A2-1.2 •	3 • 84-A46-13.1 •

Guinevere	Royal Family, The	Madam Butterfly
1987	1987	1988


BuyVintagePlates - Online

Not secure | buyvintageplates.test/portal/inventory/category/plates/create/1

BuyVintagePlates

Menu

John McDonnell

- Dashboard
- Product Inventory
- Sales & Accounting
- Order Tracking
- Reporting
- Referencing

New Sellable Product from Reference Data

Product Inventory **Create Plate Product**

Add a sellable plate product to inventory.

Winter's Peace

1981

Collection: nature's beauty

Manufacturer: allison and company

Artist: allison, betty

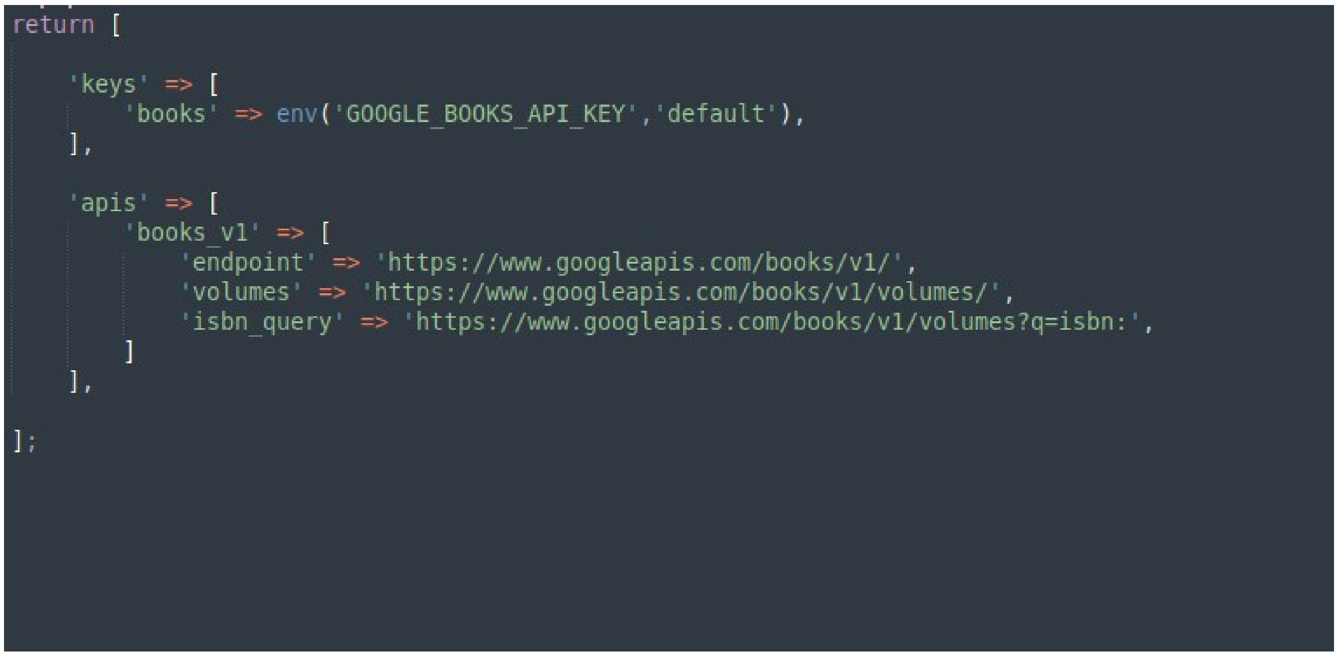
1 • 84-a2-1.1 •

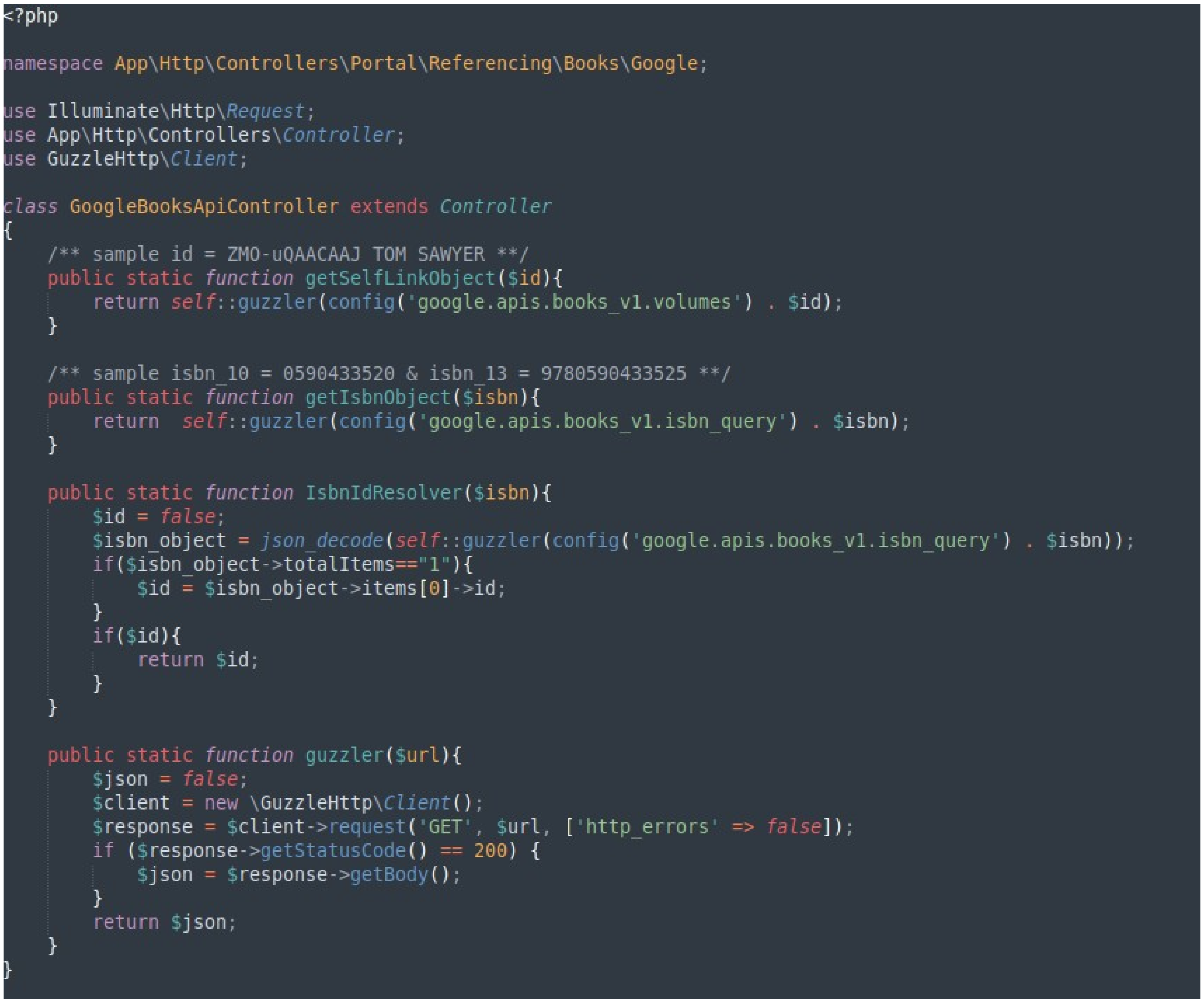
Add Product Images...

Choose File

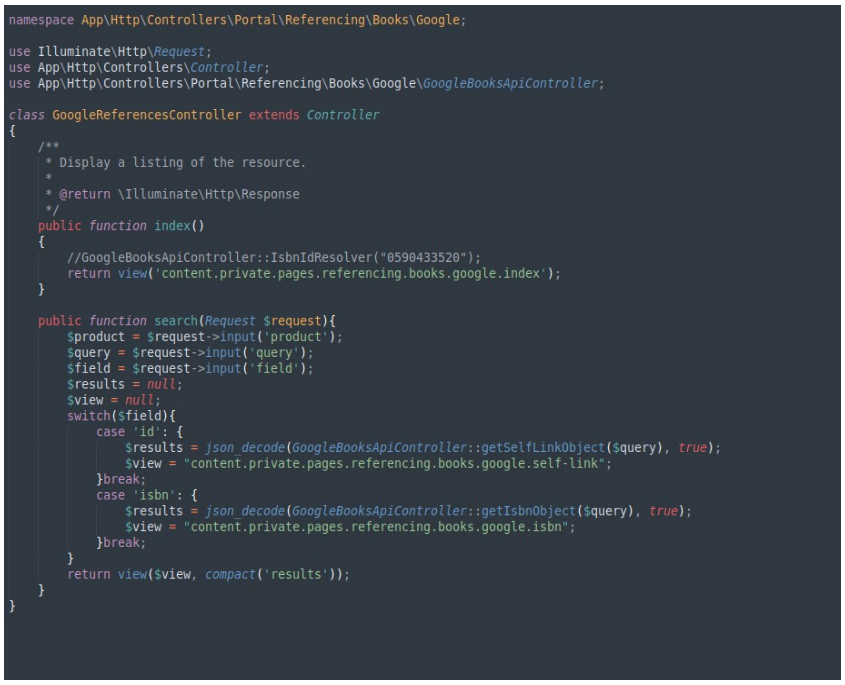
Title	winter's peace
Description	
Plate Number	
Condition	
Resale Price	
Acquistion Cost	

I also wrote the following code to integrate the google books API into laravel with custom configuration for the API endpoints, a API Interface Controller Class and an actual Portal Interface Controller class…

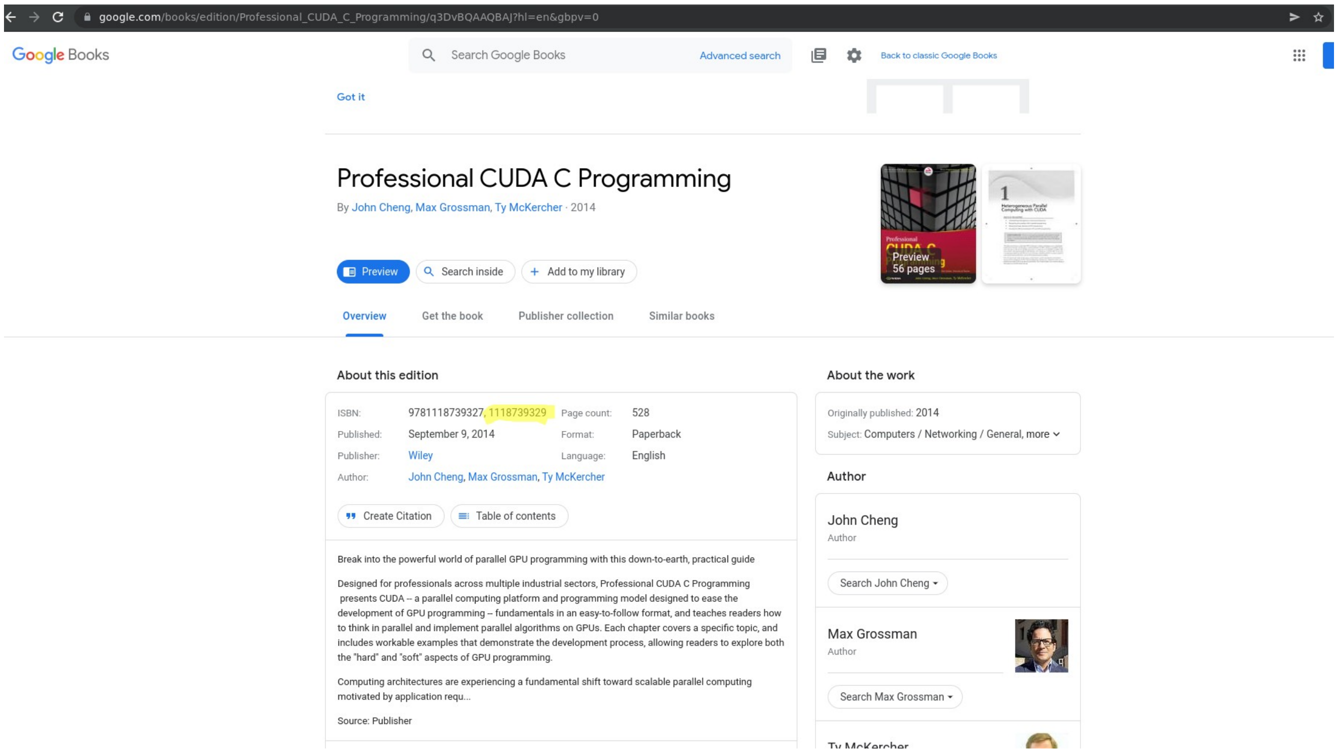

```php
return [

    'keys' => [
        'books' => env('GOOGLE_BOOKS_API_KEY','default'),
    ],

    'apis' => [
        'books_v1' => [
            'endpoint' => 'https://www.googleapis.com/books/v1/',
            'volumes' => 'https://www.googleapis.com/books/v1/volumes/',
            'isbn_query' => 'https://www.googleapis.com/books/v1/volumes?q=isbn:',
        ]
    ],

];
```



```php
<?php

namespace App\Http\Controllers\Portal\Referencing\Books\Google;

use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use GuzzleHttp\Client;

class GoogleBooksApiController extends Controller
{
    /** sample id = ZMO-uQAACAAJ TOM SAWYER **/
    public static function getSelfLinkObject($id){
        return self::guzzler(config('google.apis.books_v1.volumes') . $id);
    }

    /** sample isbn_10 = 0590433520 & isbn_13 = 9780590433525 **/
    public static function getIsbnObject($isbn){
        return  self::guzzler(config('google.apis.books_v1.isbn_query') . $isbn);
    }

    public static function IsbnIdResolver($isbn){
        $id = false;
        $isbn_object = json_decode(self::guzzler(config('google.apis.books_v1.isbn_query') . $isbn));
        if($isbn_object->totalItems=="1"){
            $id = $isbn_object->items[0]->id;
        }
        if($id){
            return $id;
        }
    }

    public static function guzzler($url){
        $json = false;
        $client = new \GuzzleHttp\Client();
        $response = $client->request('GET', $url, ['http_errors' => false]);
        if ($response->getStatusCode() == 200) {
            $json = $response->getBody();
        }
        return $json;
    }
}
```



```php
namespace App\Http\Controllers\Portal\Referencing\Books\Google;

use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use App\Http\Controllers\Portal\Referencing\Books\Google\GoogleBooksApiController;

class GoogleReferencesController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        //GoogleBooksApiController::IsbnIdResolver("0590433520");
        return view('content.private.pages.referencing.books.google.index');
    }

    public function search(Request $request){
        $product = $request->input('product');
        $query = $request->input('query');
        $field = $request->input('field');
        $results = null;
        $view = null;
        switch($field){
            case 'id': {
                $results = json_decode(GoogleBooksApiController::getSelfLinkObject($query), true);
                $view = "content.private.pages.referencing.books.google.self-link";
            }break;
            case 'isbn': {
                $results = json_decode(GoogleBooksApiController::getIsbnObject($query), true);
                $view = "content.private.pages.referencing.books.google.isbn";
            }break;
        }
        return view($view, compact('results'));
    }
}
```


google.com/books/edition/Professional_CUDA_C_Programming/q3DvBQAAQBAJ?hl=en&gbpv=0

Google Books

Search Google Books | Advanced search | Back to classic Google Books

Got it

Professional CUDA C Programming

By John Cheng, Max Grossman, Ty McKercher · 2014

Preview 56 pages

Preview | Search inside | Add to my library

Overview | Get the book | Publisher collection | Similar books

About this edition

ISBN:	9781118739327, 1118739329	Page count:	528
Published:	September 9, 2014	Format:	Paperback
Publisher:	Wiley	Language:	English
Author:	John Cheng, Max Grossman, Ty McKercher		

Create Citation | Table of contents

Break into the powerful world of parallel GPU programming with this down-to-earth, practical guide

Designed for professionals across multiple industrial sectors, Professional CUDA C Programming presents CUDA -- a parallel computing platform and programming model designed to ease the development of GPU programming -- fundamentals in an easy-to-follow format, and teaches readers how to think in parallel and implement parallel algorithms on GPUs. Each chapter covers a specific topic, and includes workable examples that demonstrate the development process, allowing readers to explore both the "hard" and "soft" aspects of GPU programming.

Computing architectures are experiencing a fundamental shift toward scalable parallel computing motivated by application requ...

Source: Publisher

About the work

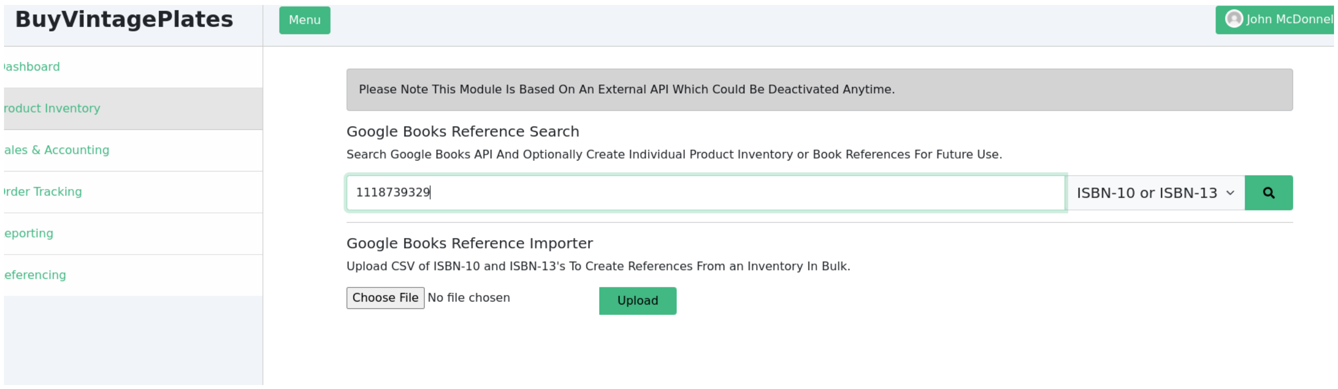
Originally published: 2014

Subject: Computers / Networking / General, more

Author

John Cheng
Author
Search John Cheng

Max Grossman
Author
Search Max Grossman


BuyVintagePlates

Menu | John McDonnel

ashboard
roduct Inventory
ales & Accounting
rder Tracking
eporting
eferencing

Please Note This Module Is Based On An External API Which Could Be Deactivated Anytime.

Google Books Reference Search

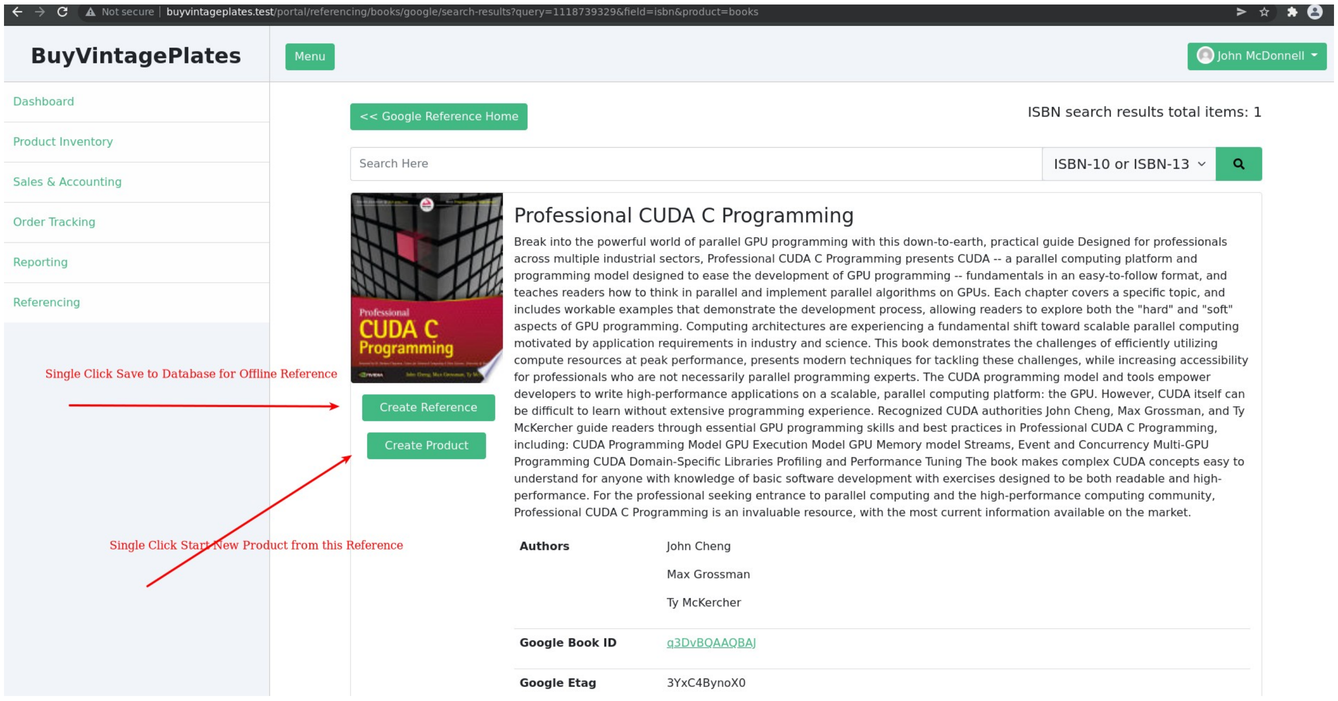
Search Google Books API And Optionally Create Individual Product Inventory or Book References For Future Use.

1118739329 | ISBN-10 or ISBN-13

Google Books Reference Importer

Upload CSV of ISBN-10 and ISBN-13's To Create References From an Inventory In Bulk.

Choose File No file chosen | Upload



Single Click Save to Database for Offline Reference

Single Click Start New Product from this Reference

Professional CUDA C Programming

Break into the powerful world of parallel GPU programming with this down-to-earth, practical guide Designed for professionals across multiple industrial sectors, Professional CUDA C Programming presents CUDA -- a parallel computing platform and programming model designed to ease the development of GPU programming -- fundamentals in an easy-to-follow format, and teaches readers how to think in parallel and implement parallel algorithms on GPUs. Each chapter covers a specific topic, and includes workable examples that demonstrate the development process, allowing readers to explore both the "hard" and "soft" aspects of GPU programming. Computing architectures are experiencing a fundamental shift toward scalable parallel computing motivated by application requirements in industry and science. This book demonstrates the challenges of efficiently utilizing compute resources at peak performance, presents modern techniques for tackling these challenges, while increasing accessibility for professionals who are not necessarily parallel programming experts. The CUDA programming model and tools empower developers to write high-performance applications on a scalable, parallel computing platform: the GPU. However, CUDA itself can be difficult to learn without extensive programming experience. Recognized CUDA authorities John Cheng, Max Grossman, and Ty McKercher guide readers through essential GPU programming skills and best practices in Professional CUDA C Programming, including: CUDA Programming Model GPU Execution Model GPU Memory model Streams, Event and Concurrency Multi-GPU Programming CUDA Domain-Specific Libraries Profiling and Performance Tuning The book makes complex CUDA concepts easy to understand for anyone with knowledge of basic software development with exercises designed to be both readable and high-performance. For the professional seeking entrance to parallel computing and the high-performance computing community, Professional CUDA C Programming is an invaluable resource, with the most current information available on the market.

Authors	John Cheng Max Grossman Ty McKercher
Google Book ID	q3DvBQAAQBAJ
Google Etag	3YxC4BynoX0