A short documentary

about product import

It’s me, Marius

Marius Strajeru

- I write code @ arnia SOFTWARE
- Magento addict
- marius.strajeru@gmail.com
- @MariusStrajeru
- http://magento.stackexchange.com/users/146/marius
- https://github.com/tzyganu

Image Sources

- https://sunspine.com
- https://www.chinabrands.com
- https://giphy.com/
- https://www.dokmart.com
- https://aionhill.com
- http://theconversation.com
- http://preparationinfo.com
- https://russelrayphotos2.com
- https://coub.com
- https://waterfm.com

The opinions expressed in this presentation belong to the author (Marius Strajeru).

This should not be used as a “best practice”.

Most of this presentation was created during the development of the import. *

* Character / class names were changed for their protection.

* Some events were exaggerated for dramatic effect.

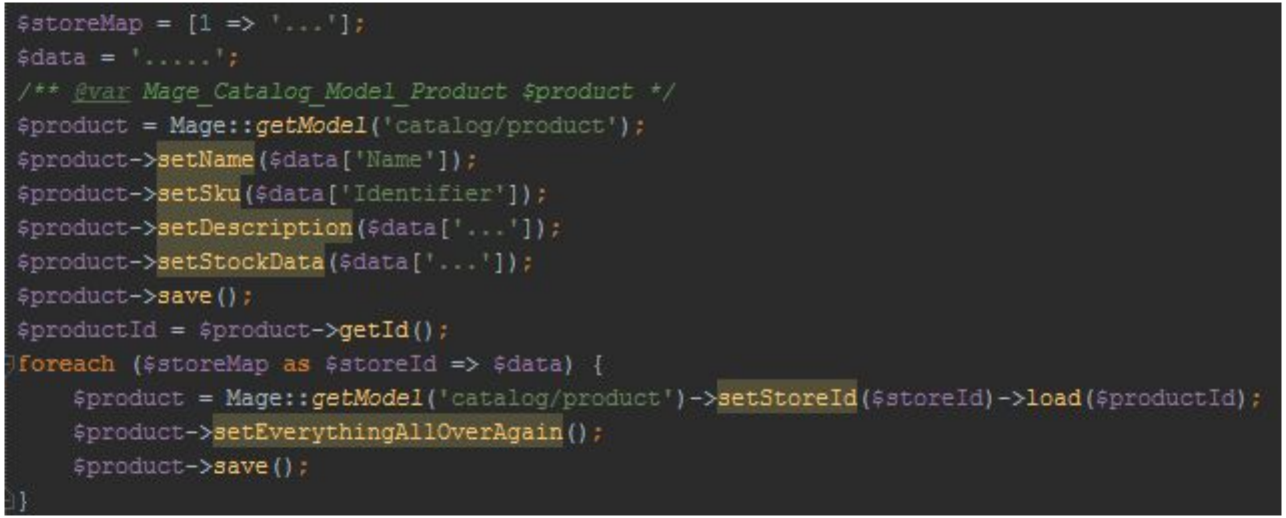
Agenda

- My first import
- Alternative solutions
- How does the core import work
- ~~How to build and import~~. How I’ve built my import
- Q&A

My First Import

My first import

- July - August 2008. - Yep. Took me around a month.
- Magento v1.0.19870.4



```
$storeMap = [1 => '...'];
$data = '.....';
/** @var Mage_Catalog_Model_Product $product */
$product = Mage::getModel('catalog/product');
$product->setName($data['Name']);
$product->setSku($data['Identifier']);
$product->setDescription($data['...']);
$product->setStockData($data['...']);
$product->save();
$productId = $product->getId();
foreach ($storeMap as $storeId => $data) {
    $product = Mage::getModel('catalog/product')->setStoreId($storeId)->load($productId);
    $product->setEverythingAllOverAgain();
    $product->save();
}
```

My first import

- Was I proud of it?

My first import

- Was I proud of it? ✓
- Did it work?

My first import

- Was I proud of it? ✓
- Did it work? ✓
- Was it fast?

My first import

- Was I proud of it? ✓
- Did it work? ✓
- Was it fast?

Alternative Solutions

Alternative solutions

Third party tools / modules

- gaMmi
- Windy om
- manage graM
- Finger toe
- trooper Him
- ….

Puzzling

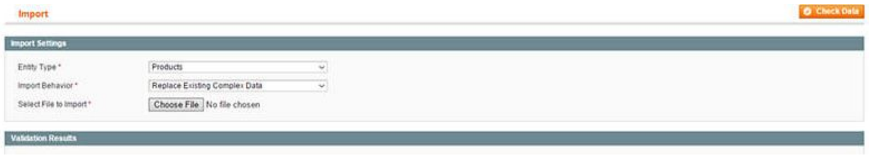
Alternative solutions

The problems

- They cost money.
- Do not fit 100% of your needs
- Not always upgrade proof.
- It’s someone else’s code so is suboptimal until proven otherwise.

The Core Import

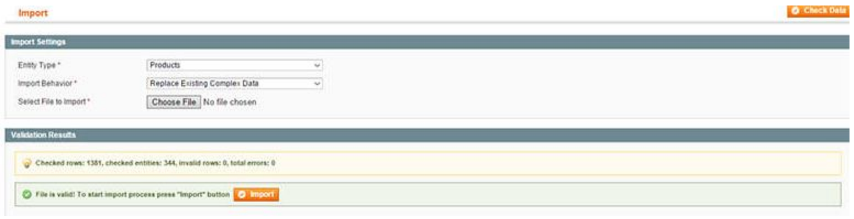
The core import



Import Settings

Entity Type *: Products
Import Behavior *: Replace Existing Complex Data
Select File to Import *: Choose File No file chosen

Validation Results



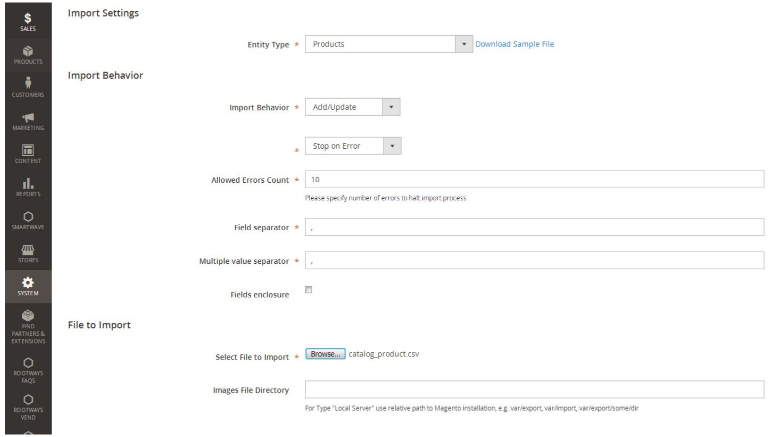
Import Settings

Entity Type *: Products
Import Behavior *: Replace Existing Complex Data
Select File to Import *: Choose File No file chosen

Validation Results

Checked rows: 1381, checked entities: 344, invalid rows: 0, total errors: 0

File is valid! To start import process press "Import" button Import



Import Settings

Entity Type *: Products — Download Sample File

Import Behavior

Import Behavior *: Add/Update
*: Stop on Error
Allowed Errors Count *: 10
Please specify number of errors to halt import process
Field separator *: ,
Multiple value separator *: ,
Fields enclosure: ☐

File to Import

Select File to Import *: Browse... catalog_product.csv
Images File Directory:
For Type "Local Server" use relative path to Magento installation, e.g. var/export, var/import, var/export/some/dir

Meet Magento™ ROMANIA RO

#MM19RO

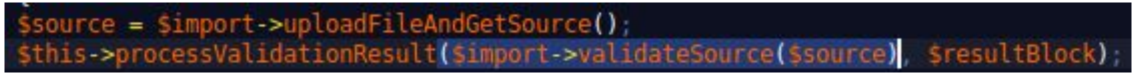
The core import - The flow

- File is uploaded
- **Contents get validated**
  - If not valid -> Show an error
  - If Valid -> Show import button
    - **Import data**

The core import

Validation

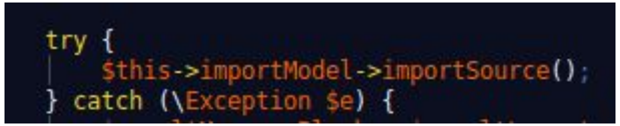
Magento\ImportExport\Controller\Adminhtml\Import\Validate::execute();



```
$source = $import->uploadFileAndGetSource();
$this->processValidationResult($import->validateSource($source), $resultBlock);
```

Import

Magento\ImportExport\Controller\Adminhtml\Import\Start::execute();



```
try {
    $this->importModel->importSource();
} catch (\Exception $e) {
```

Meet Magento ROMANIA RO

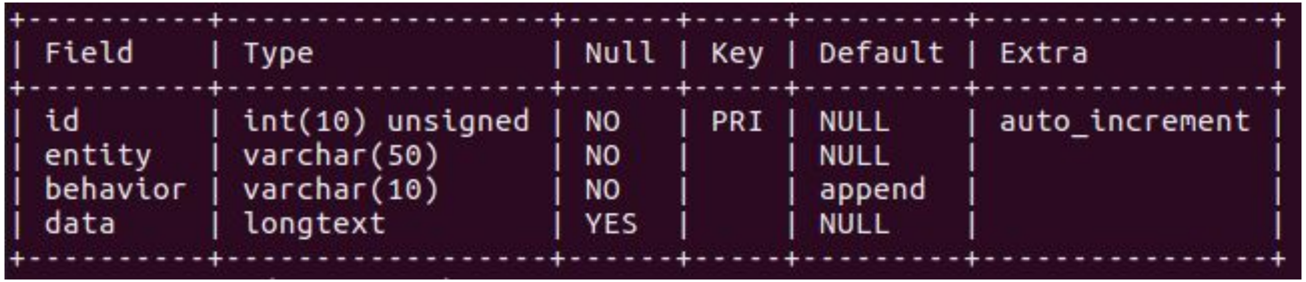
#MM19RO

The core import - Structure

- Special fields
  - sku - obviously, the SKU
  - store_view_code - store view code for which we import
  - _attribute_set, name of the attribute set of the product
  - categories - category names and paths (‘Default Category/Category1|...)
  - url_key
- Regular fields
  - attribute_code -> value
  - (multi)select attributes use option labels

The core import: Behind the scene

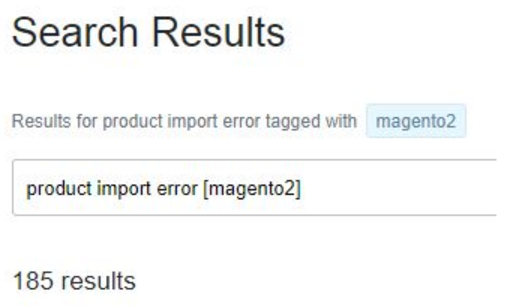
- The table `importexport_importdata`
- After validation, the table is populated with JSON encoded data from the CSV.



```
+----------+------------------+------+-----+---------+----------------+
| Field    | Type             | Null | Key | Default | Extra          |
+----------+------------------+------+-----+---------+----------------+
| id       | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| entity   | varchar(50)      | NO   |     | NULL    |                |
| behavior | varchar(10)      | NO   |     | append  |                |
| data     | longtext         | YES  |     | NULL    |                |
+----------+------------------+------+-----+---------+----------------+
```

The core import: The problems

- I don’t want to import CSV files. It’s 2019. We have APIs.
- I want to import on schedule.
- The data to import needs to be in a specific format.
- I have to create upfront the attribute options
- Errors are not clearly explained.


Search Results

Results for product import error tagged with magento2

product import error [magento2]

185 results

HOW I
DID IT

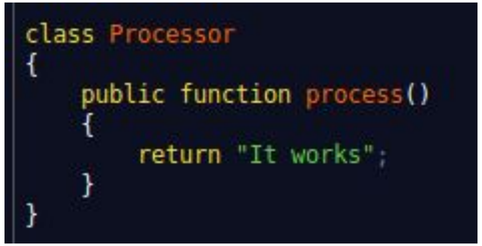
How I've built my import

- Must run via CLI and via cron (maybe something else !?!)
- Must fetch data from an url (actually more than 1).
  - Url may not return all products so 2 or more requests may be needed
- Must create missing attribute options
- Must transform data
- Must report faulty products but don't stop the import.
- Must not last 1.3 eternities.

How I’ve built my import

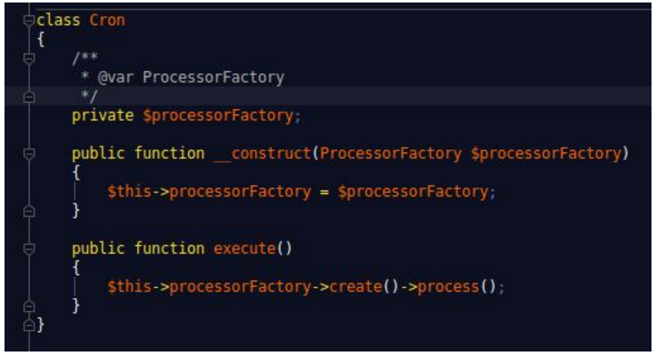
Must run via cli and cron.

Separate class



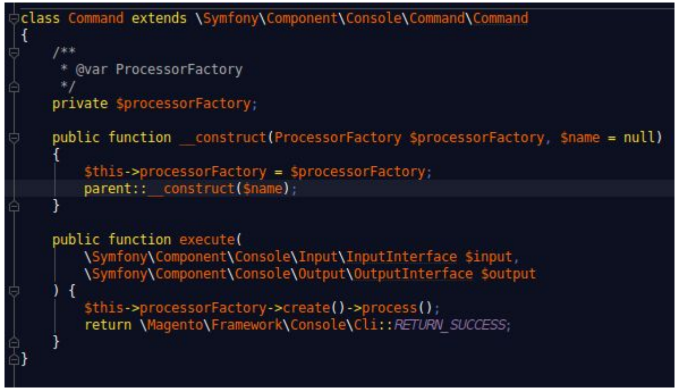
```
class Processor
{
    public function process()
    {
        return "It works";
    }
}
```

Cron class



```
class Cron
{
    /**
     * @var ProcessorFactory
     */
    private $processorFactory;

    public function __construct(ProcessorFactory $processorFactory)
    {
        $this->processorFactory = $processorFactory;
    }

    public function execute()
    {
        $this->processorFactory->create()->process();
    }
}
```

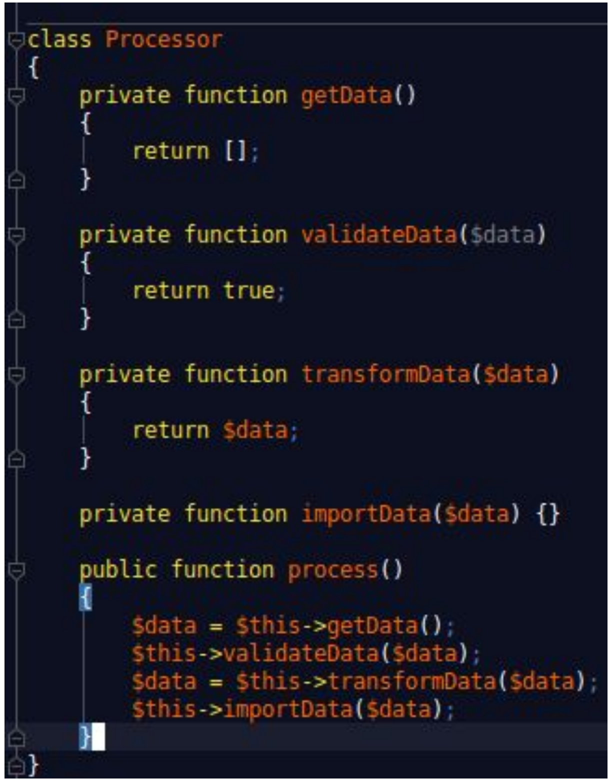
Console class



```
class Command extends \Symfony\Component\Console\Command\Command
{
    /**
     * @var ProcessorFactory
     */
    private $processorFactory;

    public function __construct(ProcessorFactory $processorFactory, $name = null)
    {
        $this->processorFactory = $processorFactory;
        parent::__construct($name);
    }

    public function execute(
        \Symfony\Component\Console\Input\InputInterface $input,
        \Symfony\Component\Console\Output\OutputInterface $output
    ) {
        $this->processorFactory->create()->process();
        return \Magento\Framework\Console\Cli::RETURN_SUCCESS;
    }
}
```

How I’ve built my import

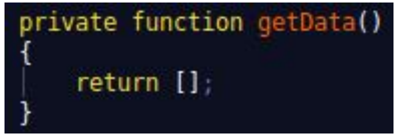
- Must fetch data from an url
- Must create missing attribute options
- Must transform data



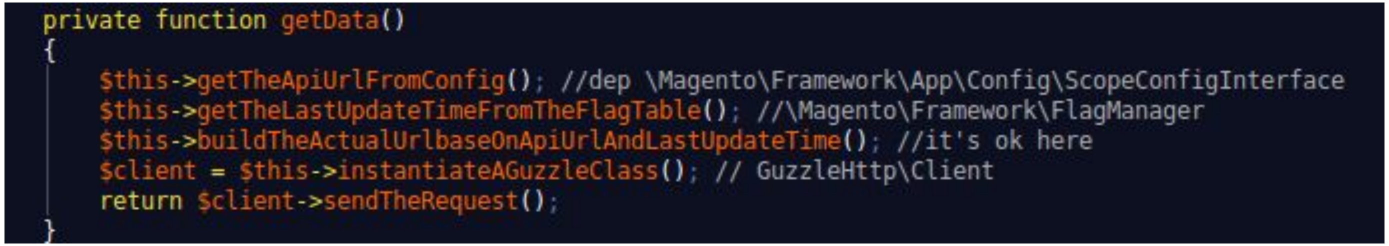
```php
class Processor
{
    private function getData()
    {
        return [];
    }

    private function validateData($data)
    {
        return true;
    }

    private function transformData($data)
    {
        return $data;
    }

    private function importData($data) {}

    public function process()
    {
        $data = $this->getData();
        $this->validateData($data);
        $data = $this->transformData($data);
        $this->importData($data);
    }
}
```

How I’ve built my import

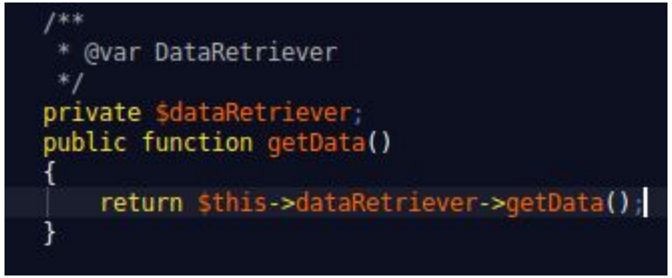
- Must fetch data from an url



```
private function getData()
{
    return [];
}
```



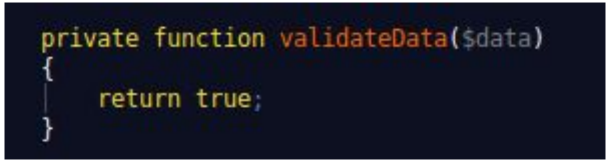
```
private function getData()
{
    $this->getTheApiUrlFromConfig(); //dep \Magento\Framework\App\Config\ScopeConfigInterface
    $this->getTheLastUpdateTimeFromTheFlagTable(); //\Magento\Framework\FlagManager
    $this->buildTheActualUrlbaseOnApiUrlAndLastUpdateTime(); //it's ok here
    $client = $this->instantiateAGuzzleClass(); // GuzzleHttp\Client
    return $client->sendTheRequest();
}
```



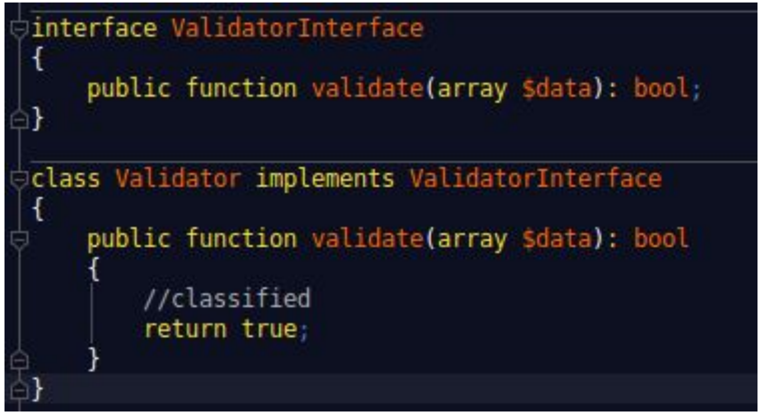
```
/**
 * @var DataRetriever
 */
private $dataRetriever;
public function getData()
{
    return $this->dataRetriever->getData();
}
```

How I’ve built my import

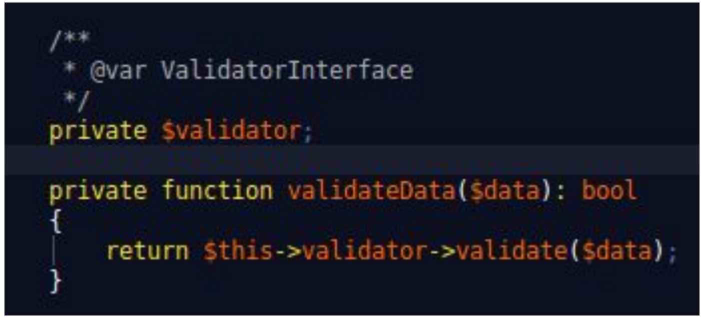
- Validate data
    - Simple and configurable are not validated the same way.
    - Maybe I can use this for stocks also?
    - Ok, separate class it is then



```php
private function validateData($data)
{
    return true;
}
```



```php
interface ValidatorInterface
{
    public function validate(array $data): bool;
}

class Validator implements ValidatorInterface
{
    public function validate(array $data): bool
    {
        //classified
        return true;
    }
}
```



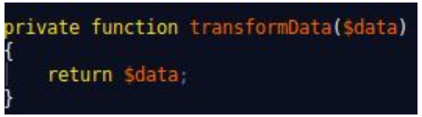
```php
/**
 * @var ValidatorInterface
 */
private $validator;

private function validateData($data): bool
{
    return $this->validator->validate($data);
}
```

Meet Magento ROMANIA RO

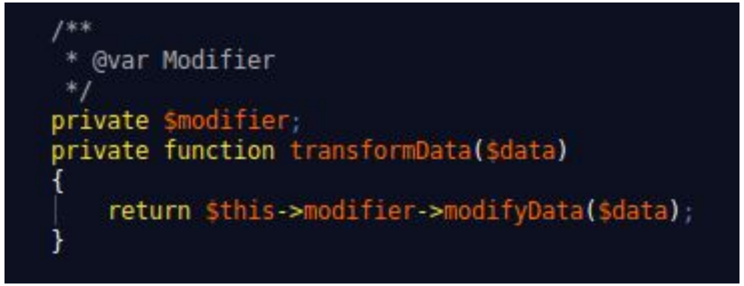
#MM19RO

How I’ve built my import

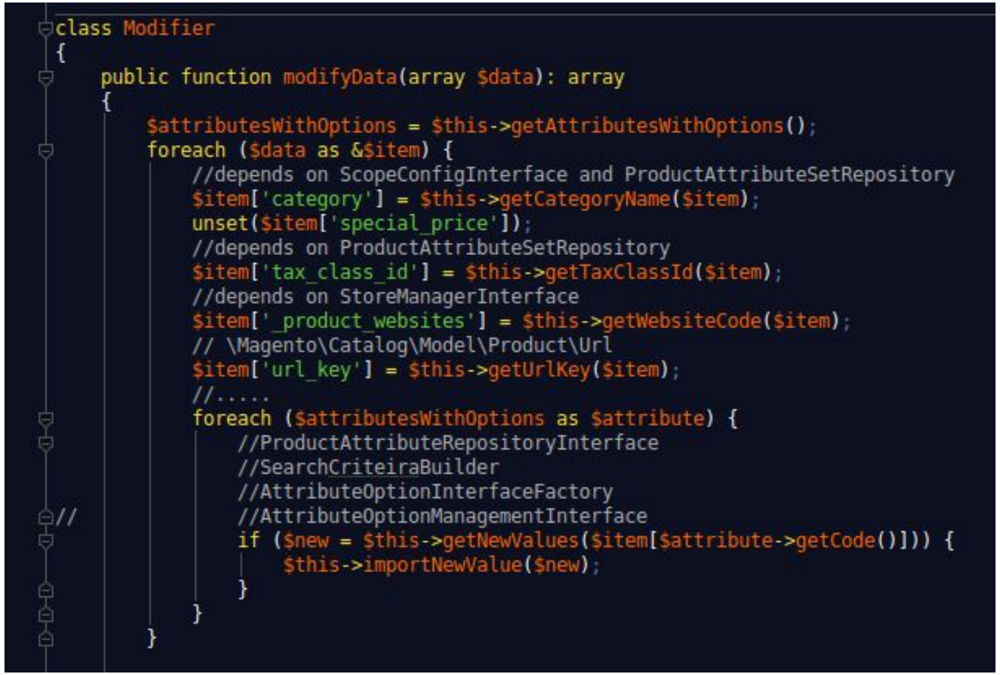
- Import missing options & transform data



```
private function transformData($data)
{
    return $data;
}
```



```
/**
 * @var Modifier
 */
private $modifier;
private function transformData($data)
{
    return $this->modifier->modifyData($data);
}
```



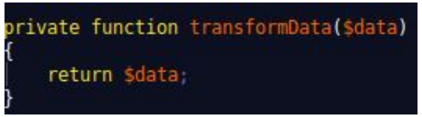
```
class Modifier
{
    public function modifyData(array $data): array
    {
        $attributesWithOptions = $this->getAttributesWithOptions();
        foreach ($data as &$item) {
            //depends on ScopeConfigInterface and ProductAttributeSetRepository
            $item['category'] = $this->getCategoryName($item);
            unset($item['special_price']);
            //depends on ProductAttributeSetRepository
            $item['tax_class_id'] = $this->getTaxClassId($item);
            //depends on StoreManagerInterface
            $item['_product_websites'] = $this->getWebsiteCode($item);
            // \Magento\Catalog\Model\Product\Url
            $item['url_key'] = $this->getUrlKey($item);
            //.....
            foreach ($attributesWithOptions as $attribute) {
                //ProductAttributeRepositoryInterface
                //SearchCriteiraBuilder
                //AttributeOptionInterfaceFactory
//              //AttributeOptionManagementInterface
                if ($new = $this->getNewValues($item[$attribute->getCode()])) {
                    $this->importNewValue($new);
                }
            }
        }
```

Meet Magento ROMANIA RO

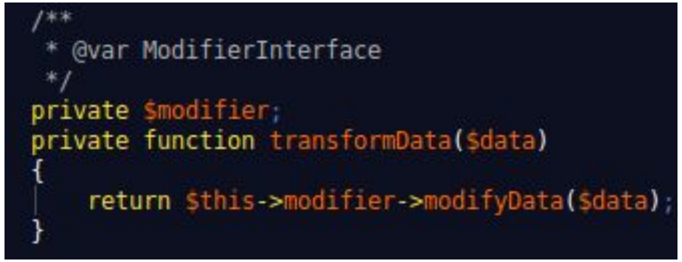
#MM19RO

How I’ve built my import

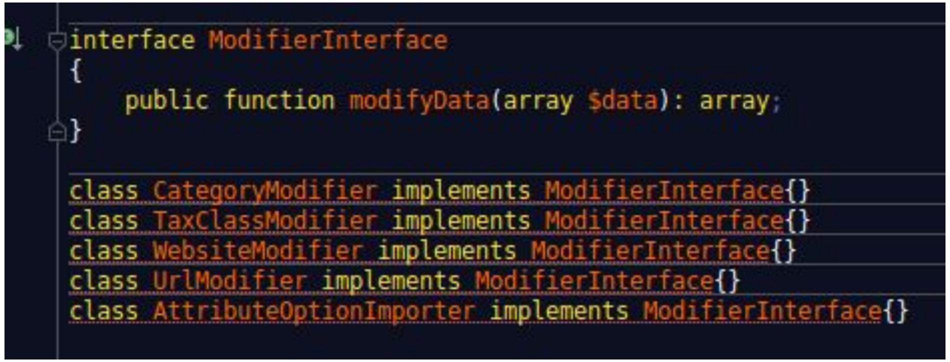
- Import missing options & transform data



```php
private function transformData($data)
{
    return $data;
}
```



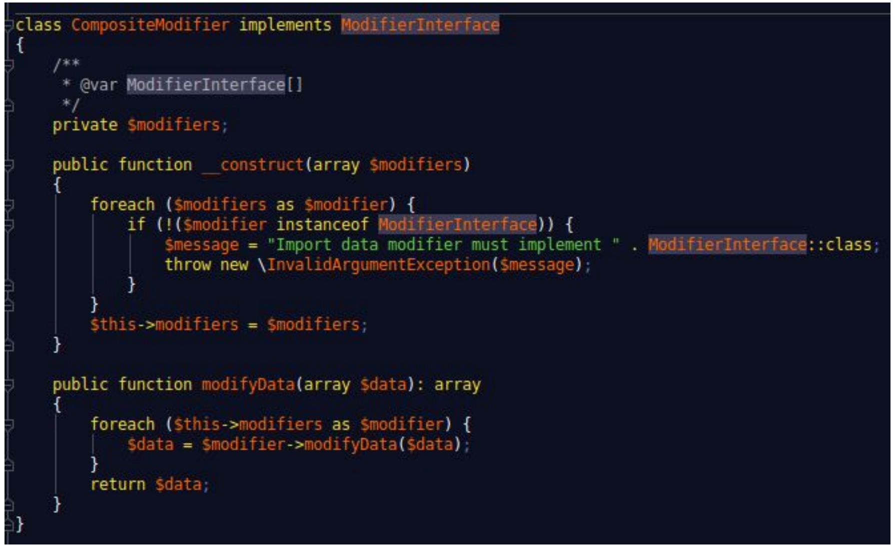
```php
/**
 * @var ModifierInterface
 */
private $modifier;
private function transformData($data)
{
    return $this->modifier->modifyData($data);
}
```



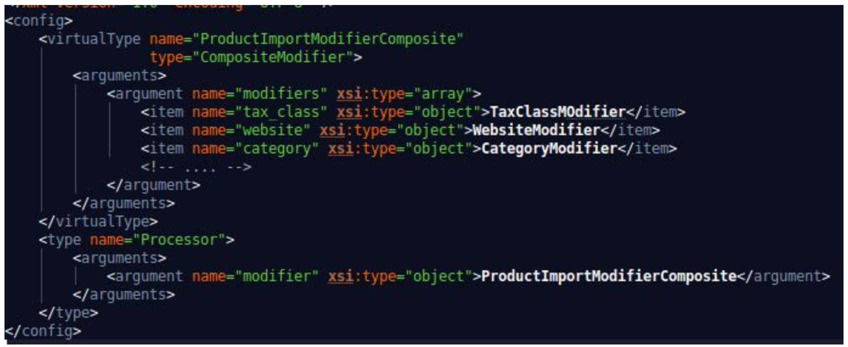
```php
interface ModifierInterface
{
    public function modifyData(array $data): array;
}

class CategoryModifier implements ModifierInterface{}
class TaxClassModifier implements ModifierInterface{}
class WebsiteModifier implements ModifierInterface{}
class UrlModifier implements ModifierInterface{}
class AttributeOptionImporter implements ModifierInterface{}
```

How I’ve built my import

- Import missing options & transform data - The composite class
- https://devdocs.magento.com/guides/v2.3/coding-standards/technical-guidelines.html



```php
class CompositeModifier implements ModifierInterface
{
    /**
     * @var ModifierInterface[]
     */
    private $modifiers;

    public function __construct(array $modifiers)
    {
        foreach ($modifiers as $modifier) {
            if (!($modifier instanceof ModifierInterface)) {
                $message = "Import data modifier must implement " . ModifierInterface::class;
                throw new \InvalidArgumentException($message);
            }
        }
        $this->modifiers = $modifiers;
    }

    public function modifyData(array $data): array
    {
        foreach ($this->modifiers as $modifier) {
            $data = $modifier->modifyData($data);
        }
        return $data;
    }
}
```

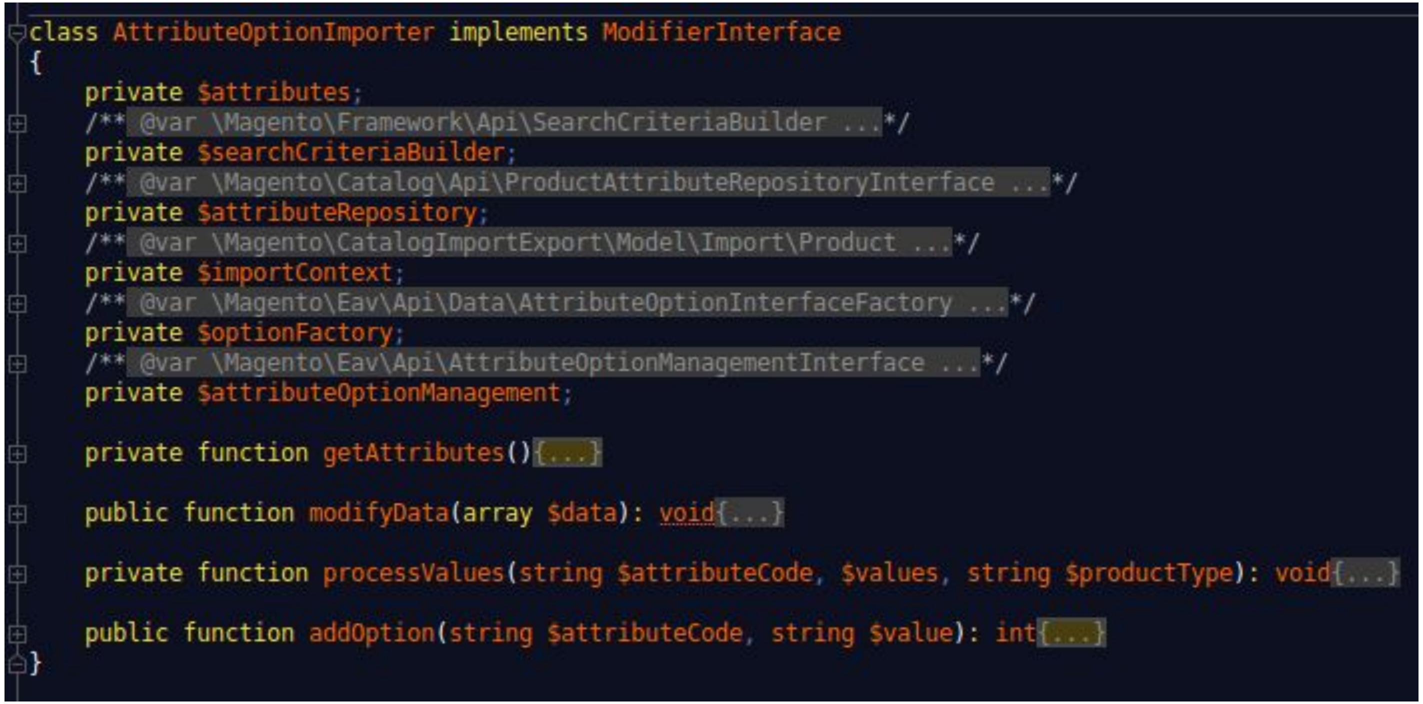


```xml
<config>
    <virtualType name="ProductImportModifierComposite"
                 type="CompositeModifier">
        <arguments>
            <argument name="modifiers" xsi:type="array">
                <item name="tax_class" xsi:type="object">TaxClassMOdifier</item>
                <item name="website" xsi:type="object">WebsiteModifier</item>
                <item name="category" xsi:type="object">CategoryModifier</item>
                <!-- .... -->
            </argument>
        </arguments>
    </virtualType>
    <type name="Processor">
        <arguments>
            <argument name="modifier" xsi:type="object">ProductImportModifierComposite</argument>
        </arguments>
    </type>
</config>
```

Meet Magento ROMANIA RO

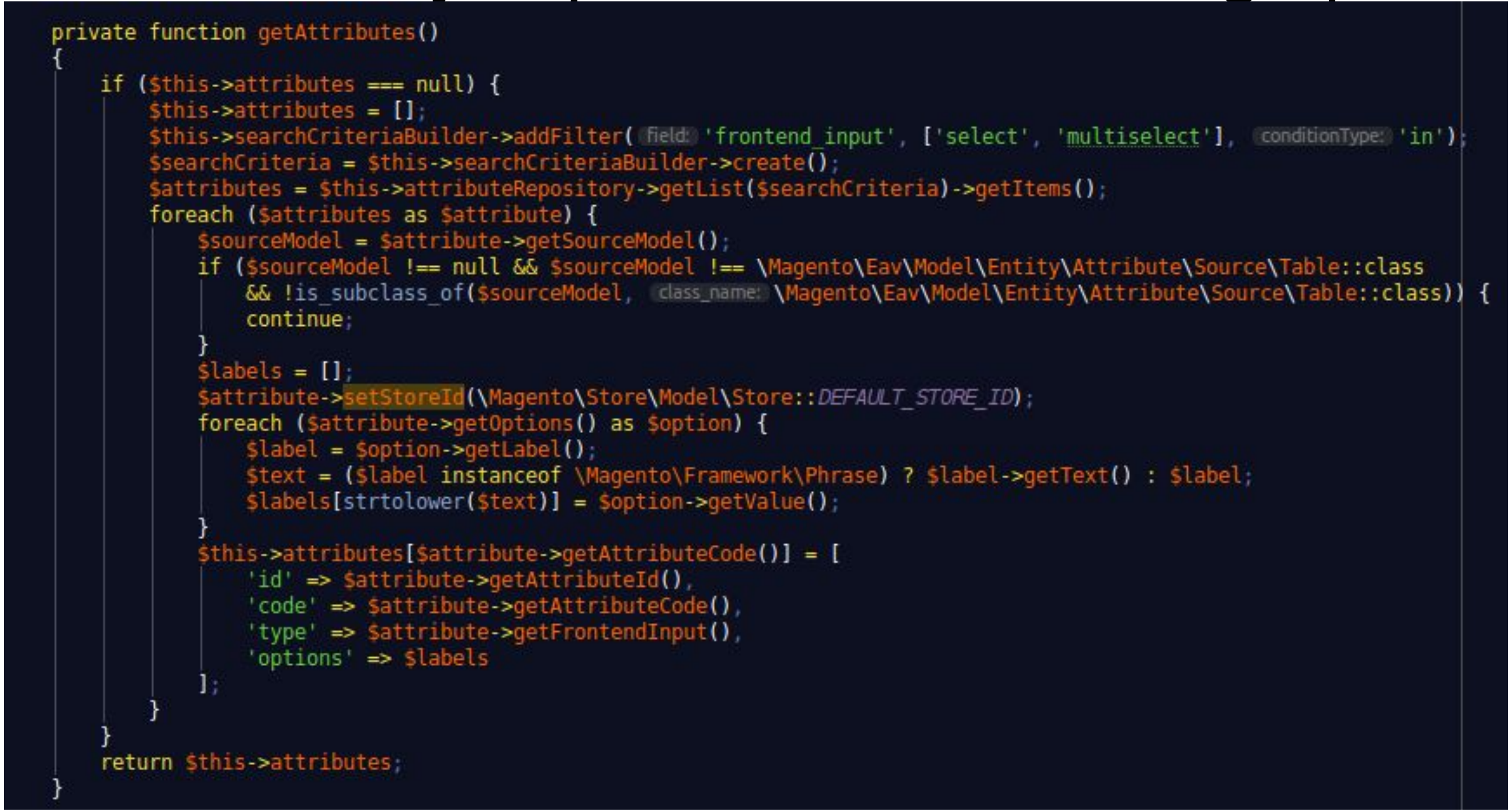
#MM19RO

How I’ve built my import - Create missing options



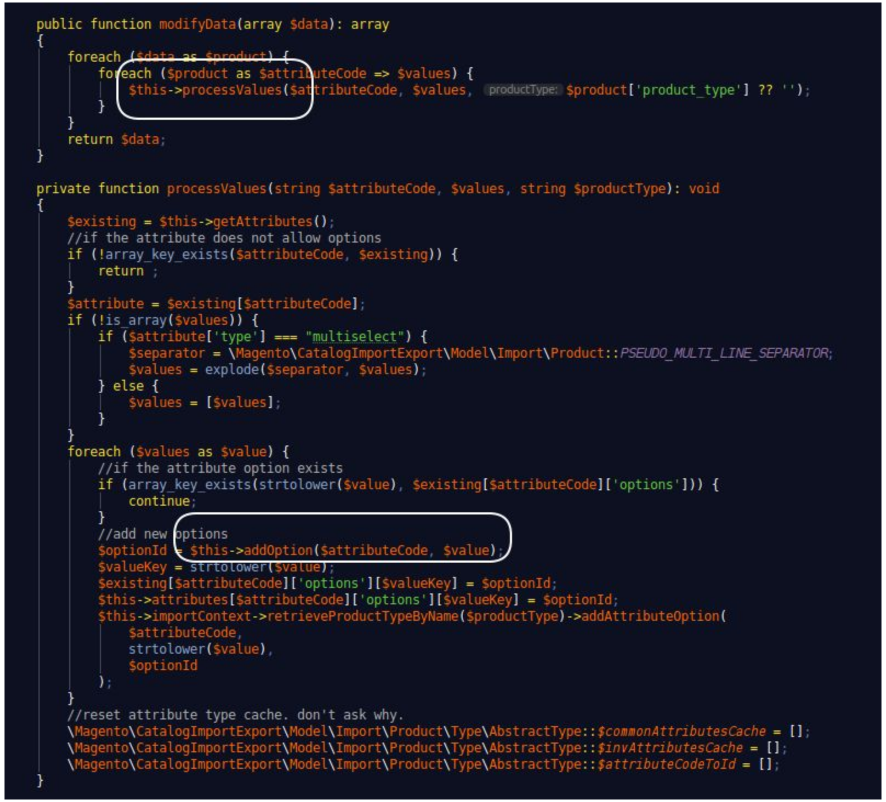
```
class AttributeOptionImporter implements ModifierInterface
{
    private $attributes;
    /** @var \Magento\Framework\Api\SearchCriteriaBuilder ...*/
    private $searchCriteriaBuilder;
    /** @var \Magento\Catalog\Api\ProductAttributeRepositoryInterface ...*/
    private $attributeRepository;
    /** @var \Magento\CatalogImportExport\Model\Import\Product ...*/
    private $importContext;
    /** @var \Magento\Eav\Api\Data\AttributeOptionInterfaceFactory ...*/
    private $optionFactory;
    /** @var \Magento\Eav\Api\AttributeOptionManagementInterface ...*/
    private $attributeOptionManagement;

    private function getAttributes(){...}

    public function modifyData(array $data): void{...}

    private function processValues(string $attributeCode, $values, string $productType): void{...}

    public function addOption(string $attributeCode, string $value): int{...}
}
```

How I’ve built my import - Create missing options



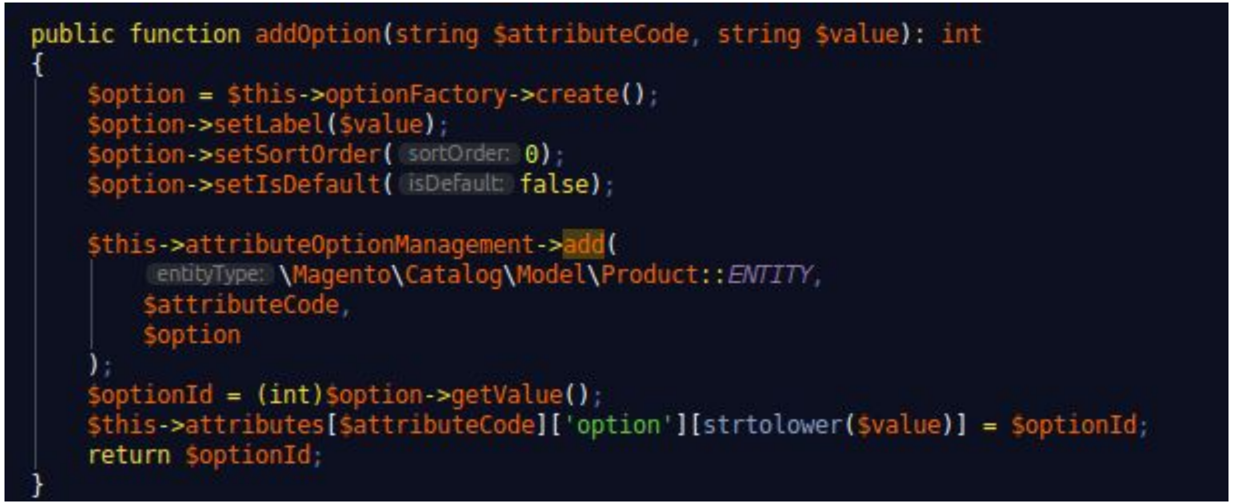
```php
private function getAttributes()
{
    if ($this->attributes === null) {
        $this->attributes = [];
        $this->searchCriteriaBuilder->addFilter( field: 'frontend_input', ['select', 'multiselect'],  conditionType: 'in');
        $searchCriteria = $this->searchCriteriaBuilder->create();
        $attributes = $this->attributeRepository->getList($searchCriteria)->getItems();
        foreach ($attributes as $attribute) {
            $sourceModel = $attribute->getSourceModel();
            if ($sourceModel !== null && $sourceModel !== \Magento\Eav\Model\Entity\Attribute\Source\Table::class
                && !is_subclass_of($sourceModel,  class_name: \Magento\Eav\Model\Entity\Attribute\Source\Table::class)) {
                continue;
            }
            $labels = [];
            $attribute->setStoreId(\Magento\Store\Model\Store::DEFAULT_STORE_ID);
            foreach ($attribute->getOptions() as $option) {
                $label = $option->getLabel();
                $text = ($label instanceof \Magento\Framework\Phrase) ? $label->getText() : $label;
                $labels[strtolower($text)] = $option->getValue();
            }
            $this->attributes[$attribute->getAttributeCode()] = [
                'id' => $attribute->getAttributeId(),
                'code' => $attribute->getAttributeCode(),
                'type' => $attribute->getFrontendInput(),
                'options' => $labels
            ];
        }
    }
    return $this->attributes;
}
```

How I’ve built my import - Create missing options



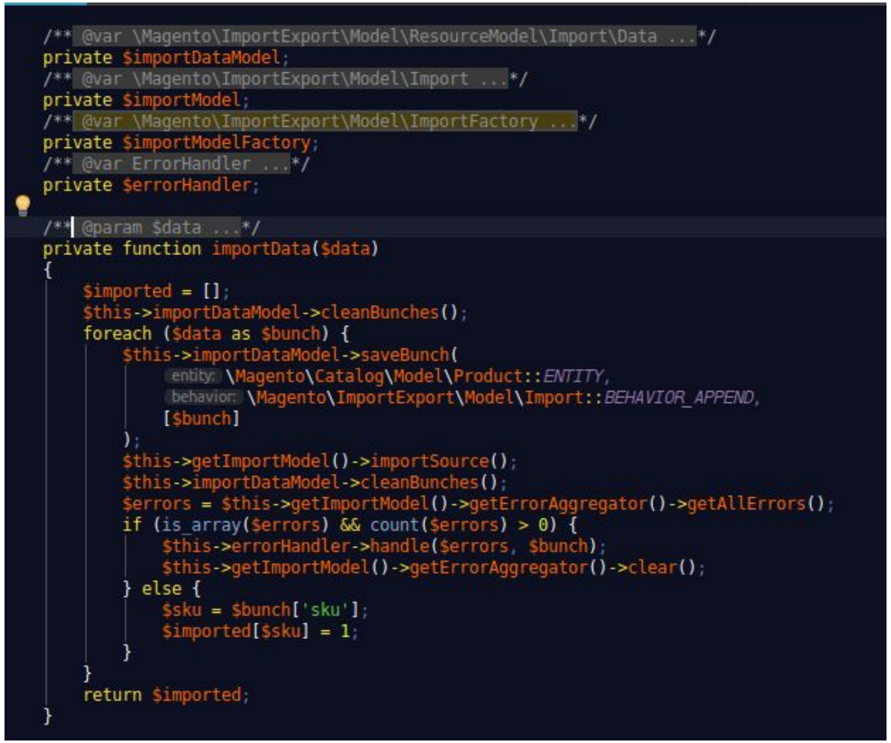
```php
public function modifyData(array $data): array
{
    foreach ($data as $product) {
        foreach ($product as $attributeCode => $values) {
            $this->processValues($attributeCode, $values,  productType: $product['product_type'] ?? '');
        }
    }
    return $data;
}

private function processValues(string $attributeCode, $values, string $productType): void
{
    $existing = $this->getAttributes();
    //if the attribute does not allow options
    if (!array_key_exists($attributeCode, $existing)) {
        return ;
    }
    $attribute = $existing[$attributeCode];
    if (!is_array($values)) {
        if ($attribute['type'] === "multiselect") {
            $separator = \Magento\CatalogImportExport\Model\Import\Product::PSEUDO_MULTI_LINE_SEPARATOR;
            $values = explode($separator, $values);
        } else {
            $values = [$values];
        }
    }
    foreach ($values as $value) {
        //if the attribute option exists
        if (array_key_exists(strtolower($value), $existing[$attributeCode]['options'])) {
            continue;
        }
        //add new options
        $optionId = $this->addOption($attributeCode, $value);
        $valueKey = strtolower($value);
        $existing[$attributeCode]['options'][$valueKey] = $optionId;
        $this->attributes[$attributeCode]['options'][$valueKey] = $optionId;
        $this->importContext->retrieveProductTypeByName($productType)->addAttributeOption(
            $attributeCode,
            strtolower($value),
            $optionId
        );
    }
    //reset attribute type cache. don't ask why.
    \Magento\CatalogImportExport\Model\Import\Product\Type\AbstractType::$commonAttributesCache = [];
    \Magento\CatalogImportExport\Model\Import\Product\Type\AbstractType::$invAttributesCache = [];
    \Magento\CatalogImportExport\Model\Import\Product\Type\AbstractType::$attributeCodeToId = [];
}
```

How I’ve built my import - Create missing options

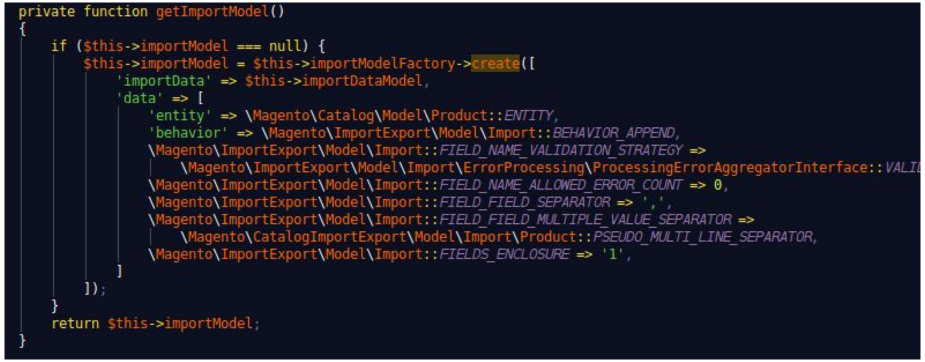


```
public function addOption(string $attributeCode, string $value): int
{
    $option = $this->optionFactory->create();
    $option->setLabel($value);
    $option->setSortOrder( sortOrder: 0);
    $option->setIsDefault( isDefault: false);

    $this->attributeOptionManagement->add(
        entityType: \Magento\Catalog\Model\Product::ENTITY,
        $attributeCode,
        $option
    );
    $optionId = (int)$option->getValue();
    $this->attributes[$attributeCode]['option'][strtolower($value)] = $optionId;
    return $optionId;
}
```

How I’ve built my import - The actual import



```php
/** @var \Magento\ImportExport\Model\ResourceModel\Import\Data ...*/
private $importDataModel;
/** @var \Magento\ImportExport\Model\Import ...*/
private $importModel;
/** @var \Magento\ImportExport\Model\ImportFactory ...*/
private $importModelFactory;
/** @var ErrorHandler ...*/
private $errorHandler;

/** @param $data ...*/
private function importData($data)
{
    $imported = [];
    $this->importDataModel->cleanBunches();
    foreach ($data as $bunch) {
        $this->importDataModel->saveBunch(
            entity: \Magento\Catalog\Model\Product::ENTITY,
            behavior: \Magento\ImportExport\Model\Import::BEHAVIOR_APPEND,
            [$bunch]
        );
        $this->getImportModel()->importSource();
        $this->importDataModel->cleanBunches();
        $errors = $this->getImportModel()->getErrorAggregator()->getAllErrors();
        if (is_array($errors) && count($errors) > 0) {
            $this->errorHandler->handle($errors, $bunch);
            $this->getImportModel()->getErrorAggregator()->clear();
        } else {
            $sku = $bunch['sku'];
            $imported[$sku] = 1;
        }
    }
    return $imported;
}
```

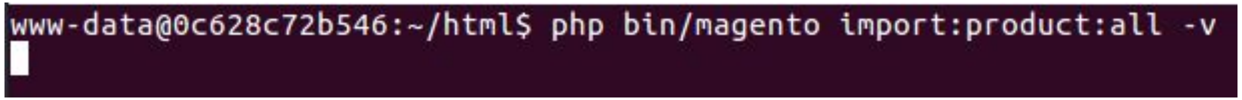


```php
private function getImportModel()
{
    if ($this->importModel === null) {
        $this->importModel = $this->importModelFactory->create([
            'importData' => $this->importDataModel,
            'data' => [
                'entity' => \Magento\Catalog\Model\Product::ENTITY,
                'behavior' => \Magento\ImportExport\Model\Import::BEHAVIOR_APPEND,
                \Magento\ImportExport\Model\Import::FIELD_NAME_VALIDATION_STRATEGY =>
                    \Magento\ImportExport\Model\Import\ErrorProcessing\ProcessingErrorAggregatorInterface::VALI
                \Magento\ImportExport\Model\Import::FIELD_NAME_ALLOWED_ERROR_COUNT => 0,
                \Magento\ImportExport\Model\Import::FIELD_FIELD_SEPARATOR => ',',
                \Magento\ImportExport\Model\Import::FIELD_FIELD_MULTIPLE_VALUE_SEPARATOR =>
                    \Magento\CatalogImportExport\Model\Import\Product::PSEUDO_MULTI_LINE_SEPARATOR,
                \Magento\ImportExport\Model\Import::FIELDS_ENCLOSURE => '1',
            ]
        ]);
    }
    return $this->importModel;
}
```

Meet Magento ROMANIA RO

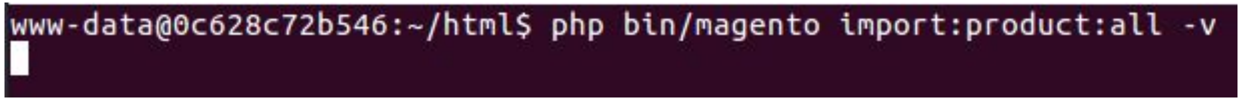
#MM19RO

How I’ve built my import - Let’s try it

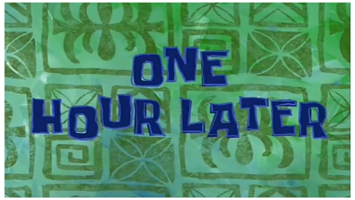


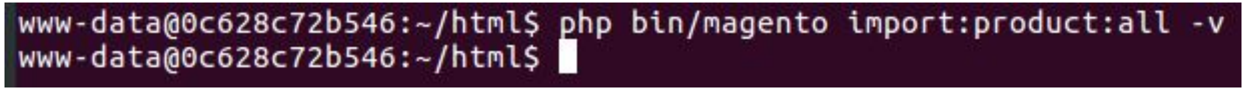
```
www-data@0c628c72b546:~/html$ php bin/magento import:product:all -v
```

How I’ve built my import - It works



```
www-data@0c628c72b546:~/html$ php bin/magento import:product:all -v
█
```



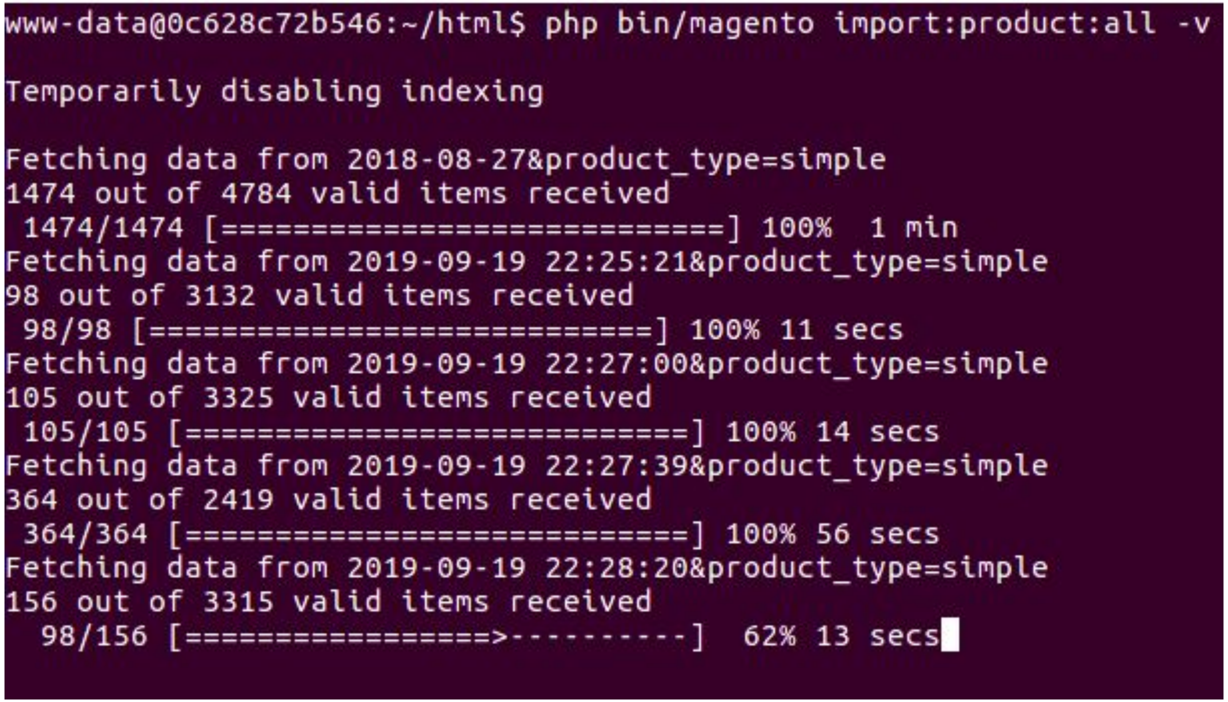


```
www-data@0c628c72b546:~/html$ php bin/magento import:product:all -v
www-data@0c628c72b546:~/html$ █
```

How I’ve built my import - It works

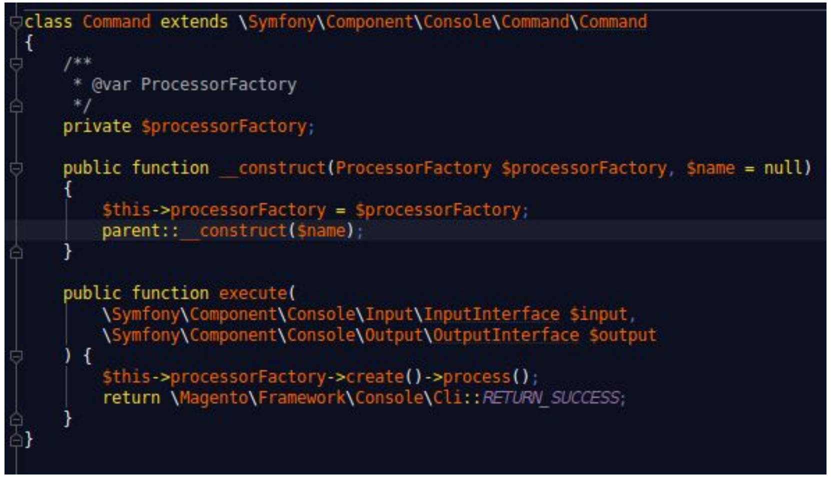
- 3 websites (1 store view each)
- 16173 products imported
    - 13313 simple
    - 2860 configurable
- 2 categories
- 388 attributes
- 4618 attribute options created
- 0 images
- Roughly 1 hour
- ….on a dev machine

How I’ve built my import - This would be nicer

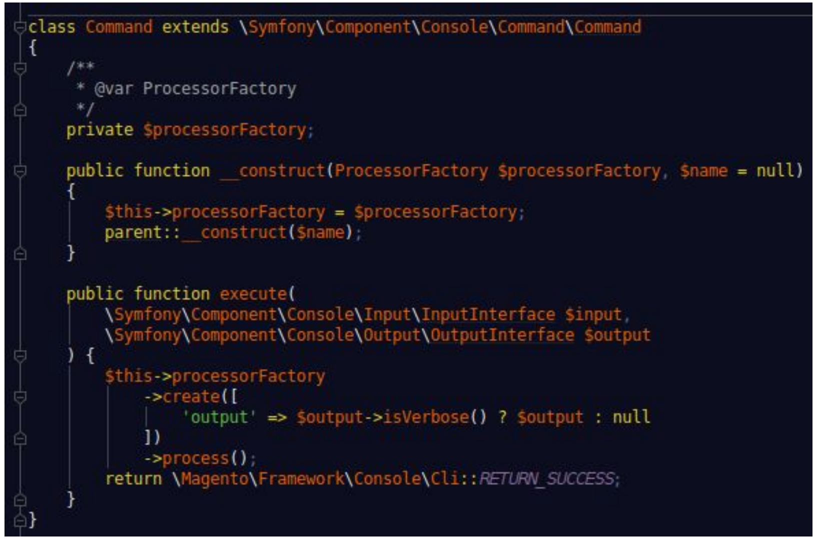


```
www-data@0c628c72b546:~/html$ php bin/magento import:product:all -v

Temporarily disabling indexing

Fetching data from 2018-08-27&product_type=simple
1474 out of 4784 valid items received
 1474/1474 [============================] 100%  1 min
Fetching data from 2019-09-19 22:25:21&product_type=simple
98 out of 3132 valid items received
 98/98 [============================] 100% 11 secs
Fetching data from 2019-09-19 22:27:00&product_type=simple
105 out of 3325 valid items received
 105/105 [============================] 100% 14 secs
Fetching data from 2019-09-19 22:27:39&product_type=simple
364 out of 2419 valid items received
 364/364 [============================] 100% 56 secs
Fetching data from 2019-09-19 22:28:20&product_type=simple
156 out of 3315 valid items received
  98/156 [=================>----------]  62% 13 secs
```

How I've built my import - Beautifying it

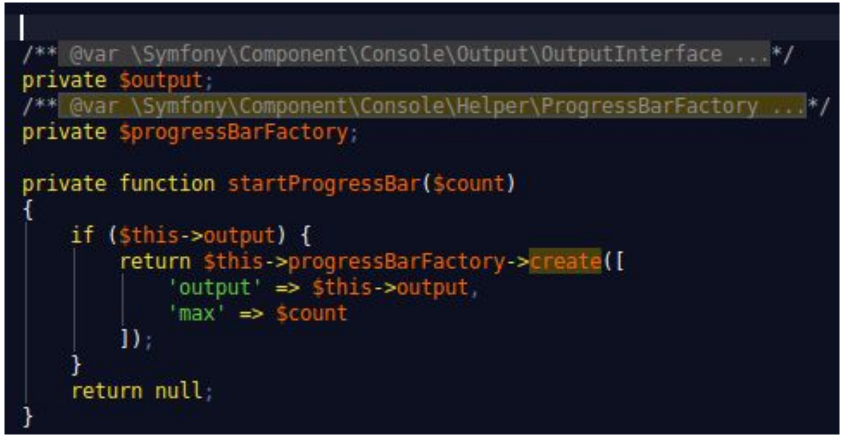


```
class Command extends \Symfony\Component\Console\Command\Command
{
    /**
     * @var ProcessorFactory
     */
    private $processorFactory;

    public function __construct(ProcessorFactory $processorFactory, $name = null)
    {
        $this->processorFactory = $processorFactory;
        parent::__construct($name);
    }

    public function execute(
        \Symfony\Component\Console\Input\InputInterface $input,
        \Symfony\Component\Console\Output\OutputInterface $output
    ) {
        $this->processorFactory->create()->process();
        return \Magento\Framework\Console\Cli::RETURN_SUCCESS;
    }
}
```

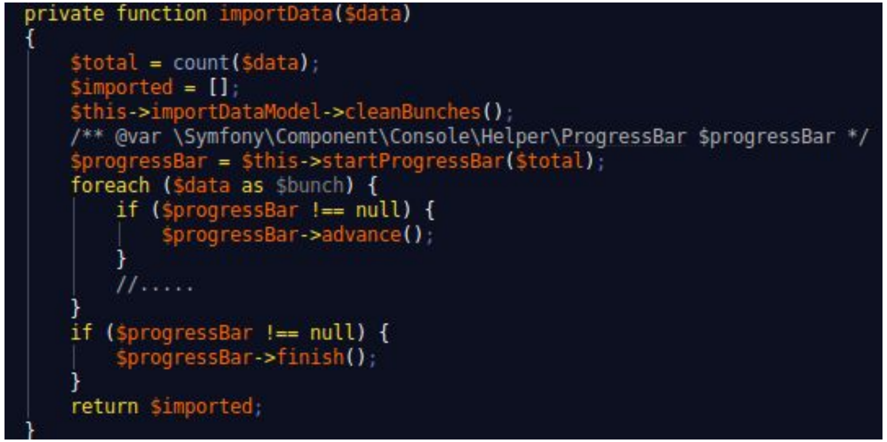


```
class Command extends \Symfony\Component\Console\Command\Command
{
    /**
     * @var ProcessorFactory
     */
    private $processorFactory;

    public function __construct(ProcessorFactory $processorFactory, $name = null)
    {
        $this->processorFactory = $processorFactory;
        parent::__construct($name);
    }

    public function execute(
        \Symfony\Component\Console\Input\InputInterface $input,
        \Symfony\Component\Console\Output\OutputInterface $output
    ) {
        $this->processorFactory
            ->create([
                'output' => $output->isVerbose() ? $output : null
            ])
            ->process();
        return \Magento\Framework\Console\Cli::RETURN_SUCCESS;
    }
}
```

How I’ve built my import - Beautifying it

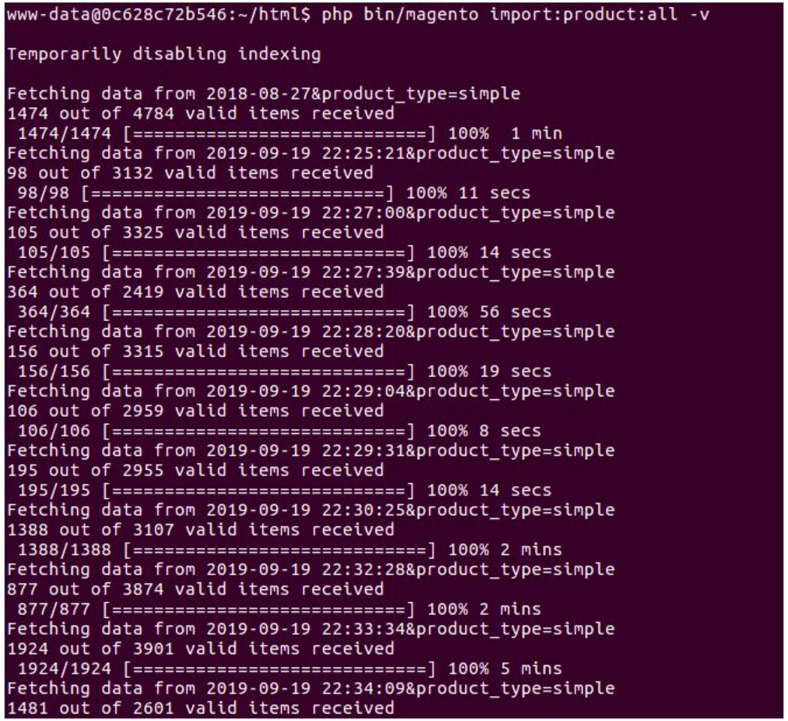


```
/** @var \Symfony\Component\Console\Output\OutputInterface ...*/
private $output;
/** @var \Symfony\Component\Console\Helper\ProgressBarFactory ...*/
private $progressBarFactory;

private function startProgressBar($count)
{
    if ($this->output) {
        return $this->progressBarFactory->create([
            'output' => $this->output,
            'max' => $count
        ]);
    }
    return null;
}
```

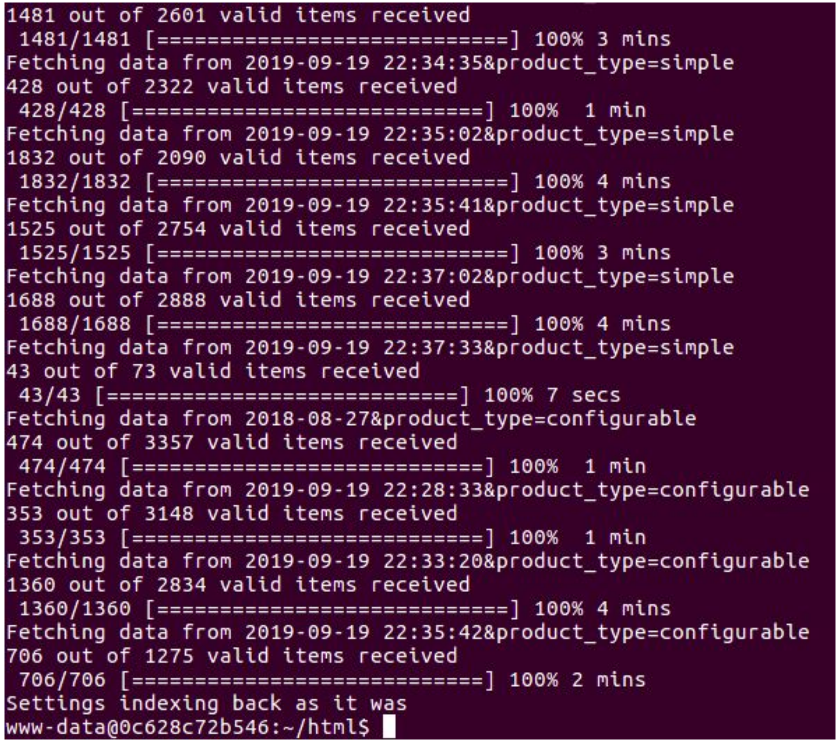


```
private function importData($data)
{
    $total = count($data);
    $imported = [];
    $this->importDataModel->cleanBunches();
    /** @var \Symfony\Component\Console\Helper\ProgressBar $progressBar */
    $progressBar = $this->startProgressBar($total);
    foreach ($data as $bunch) {
        if ($progressBar !== null) {
            $progressBar->advance();
        }
        //.....
    }
    if ($progressBar !== null) {
        $progressBar->finish();
    }
    return $imported;
}
```

How I’ve built my import - Result



```
www-data@0c628c72b546:~/html$ php bin/magento import:product:all -v

Temporarily disabling indexing

Fetching data from 2018-08-27&product_type=simple
1474 out of 4784 valid items received
 1474/1474 [============================] 100%  1 min
Fetching data from 2019-09-19 22:25:21&product_type=simple
98 out of 3132 valid items received
 98/98 [============================] 100% 11 secs
Fetching data from 2019-09-19 22:27:00&product_type=simple
105 out of 3325 valid items received
 105/105 [============================] 100% 14 secs
Fetching data from 2019-09-19 22:27:39&product_type=simple
364 out of 2419 valid items received
 364/364 [============================] 100% 56 secs
Fetching data from 2019-09-19 22:28:20&product_type=simple
156 out of 3315 valid items received
 156/156 [============================] 100% 19 secs
Fetching data from 2019-09-19 22:29:04&product_type=simple
106 out of 2959 valid items received
 106/106 [============================] 100% 8 secs
Fetching data from 2019-09-19 22:29:31&product_type=simple
195 out of 2955 valid items received
 195/195 [============================] 100% 14 secs
Fetching data from 2019-09-19 22:30:25&product_type=simple
1388 out of 3107 valid items received
 1388/1388 [============================] 100% 2 mins
Fetching data from 2019-09-19 22:32:28&product_type=simple
877 out of 3874 valid items received
 877/877 [============================] 100% 2 mins
Fetching data from 2019-09-19 22:33:34&product_type=simple
1924 out of 3901 valid items received
 1924/1924 [============================] 100% 5 mins
Fetching data from 2019-09-19 22:34:09&product_type=simple
1481 out of 2601 valid items received
```



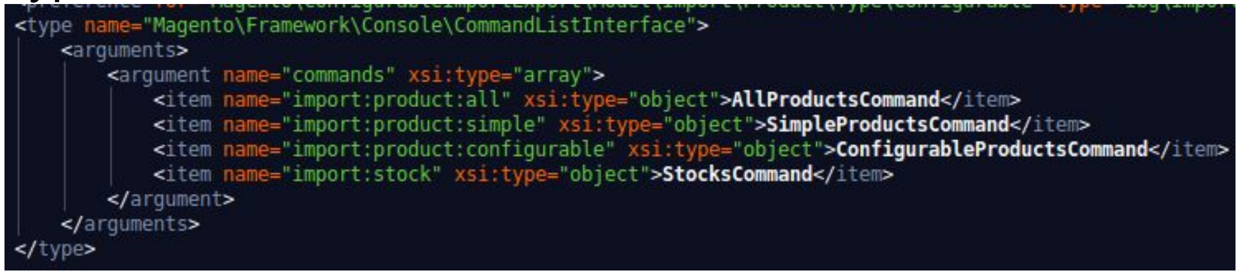
```
1481 out of 2601 valid items received
 1481/1481 [============================] 100% 3 mins
Fetching data from 2019-09-19 22:34:35&product_type=simple
428 out of 2322 valid items received
 428/428 [============================] 100%  1 min
Fetching data from 2019-09-19 22:35:02&product_type=simple
1832 out of 2090 valid items received
 1832/1832 [============================] 100% 4 mins
Fetching data from 2019-09-19 22:35:41&product_type=simple
1525 out of 2754 valid items received
 1525/1525 [============================] 100% 3 mins
Fetching data from 2019-09-19 22:37:02&product_type=simple
1688 out of 2888 valid items received
 1688/1688 [============================] 100% 4 mins
Fetching data from 2019-09-19 22:37:33&product_type=simple
43 out of 73 valid items received
 43/43 [============================] 100% 7 secs
Fetching data from 2018-08-27&product_type=configurable
474 out of 3357 valid items received
 474/474 [============================] 100%  1 min
Fetching data from 2019-09-19 22:28:33&product_type=configurable
353 out of 3148 valid items received
 353/353 [============================] 100%  1 min
Fetching data from 2019-09-19 22:33:20&product_type=configurable
1360 out of 2834 valid items received
 1360/1360 [============================] 100% 4 mins
Fetching data from 2019-09-19 22:35:42&product_type=configurable
706 out of 1275 valid items received
 706/706 [============================] 100% 2 mins
Settings indexing back as it was
www-data@0c628c72b546:~/html$
```

How I’ve built my import - small stuff

- During the import the indexing is faked to “on schedule”
- Events are dispatched when import is over
- Difference between simple & configurable imports is done via virtual types.
- To import all products a CompositeProcessor is created.
- Surprise… Stock import works just by configuring virtual types
    - … ok! one extra validator class

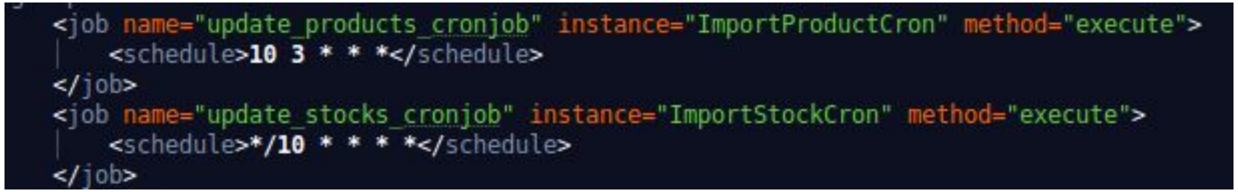
How I've built my import - Lessons learned

- Core import is a good tool (may require additional codding).
- Composite Classes are awesome.
- Proper URL rewrites handling is hard
- Virtual types can be console commands



```xml
<type name="Magento\Framework\Console\CommandListInterface">
    <arguments>
        <argument name="commands" xsi:type="array">
            <item name="import:product:all" xsi:type="object">AllProductsCommand</item>
            <item name="import:product:simple" xsi:type="object">SimpleProductsCommand</item>
            <item name="import:product:configurable" xsi:type="object">ConfigurableProductsCommand</item>
            <item name="import:stock" xsi:type="object">StocksCommand</item>
        </argument>
    </arguments>
</type>
```

- … and crons



```xml
<job name="update_products_cronjob" instance="ImportProductCron" method="execute">
    <schedule>10 3 * * *</schedule>
</job>
<job name="update_stocks_cronjob" instance="ImportStockCron" method="execute">
    <schedule>*/10 * * * *</schedule>
</job>
```

Q
&
A