Porting A Complex Extension To Magento 2

A case study at integer_net

DevelopersParadise

2016 / Opatija / Croatia

Fabian Schmengler

Magento Backend Developer – integer_net

DevelopersParadise

2016 / Opatija / Croatia

Subject: IntegerNet_Solr

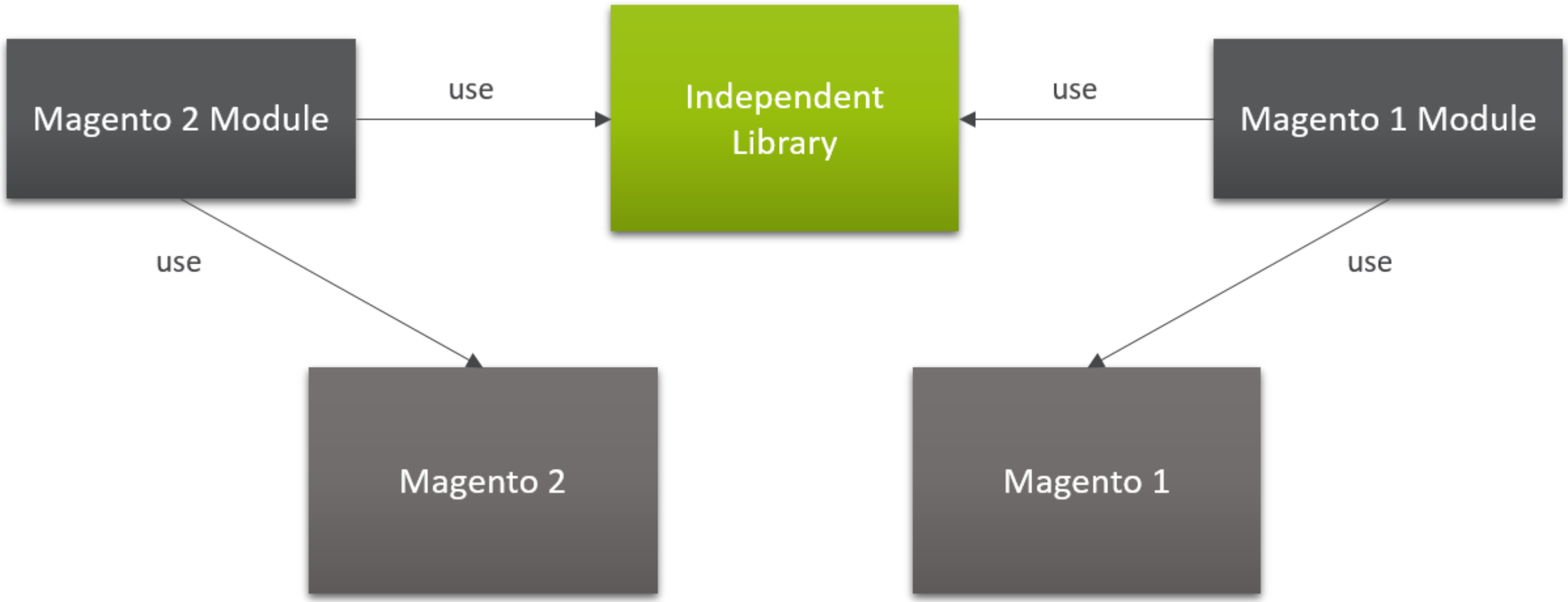
- Main Features
  - Replaces Search and Categories
  - Layered Navigation
  - Search Suggestions

- Size (without tests)
  - 2304 Logical Lines of Code (LLOC)
  - 79 Classes, 3 Interfaces
  - 768x „Mage“

Structured Rewrite For Magento 2

- Step 1: Decouple Business Logic
- Step 2: Port Remaining Module Code

Step 1: Decouple Business Logic



Before Refactoring

- Add missing **integration tests**!
  - EcomDev_PHPUnit
- (having functional tests is great too)

This is NOT optional!

Make changes with confidence

Refactoring: Easy Start

~~Mage::~~

- Start with classes with least interaction with Magento. Eliminate dependencies.

```
Mage::throwException($msg)      => throw new IntegerNet_Solr_Exception($msg)

Mage::getStoreConfig($path)     =>  (inject config value object)

    final class IntegerNet_Solr_Config_Server
    {
        private $host, $port, ...;
        public function __construct($host, $port, ...) { ... }
        public function getHost() { ... }
        public function getPort() { ... }
        ...
    }
```


Refactoring: Interface Segregation

- **Example:** Helper with different groups of methods
  1. Create interfaces with ex. methods
  2. Let Helper implement the interfaces
  3. Require interfaces instead of helper
  4. Inject helper

~~`Mage::helper()`~~

Refactoring: Split Big Classes

- Extract methods that don't use any mutable attributes of `$this` to other classes
- Extract all mutable state, grouped with the methods operating on this state

Divide and conquer!

Refactoring: Split Big Classes

- If used in many places:
  - Refactor to Facade
  - Keep interface, delegate
- Example: Big „Result“ singleton
  - **Before:** 10 public methods, 221 LLOC
  - **After:** 6 public methods, 31 LLOC

Rebuild Components

```
namespace IntegerNet\Solr\Query;

final class SearchString
{
    /**
     * @var $rawString string
     */
    private $rawString;
    /**
     * @var $escapedString string
     */
    private $escapedString
    public function __construct($rawString)
    {
        $this->rawString = $rawString;
    }

    public function getRawString()
        return $this->rawString;
    }

    public function getEscapedString()
    {
        if ($this->escapedString === null) {
            $this->escapedString = $this->escape($this->getRawString());
        }
        return $this->escapedString;
```

- Create new structures bottom up if:
  - Concept does not exist yet
  - Scattered too much in original code
- Example: „Query“
  - Helper to escape query strings
  - Search query model for synonyms
  - Query parameters, already extracted to lib

Plan ➔ develop small independent units (TDD!) ➔ replace old implementation

Decouple Magento Models

```
interface ProductRepository
{
    /**
     * Return product iterator, which may implement lazy loading
     *
     * @return ProductIterator
     */
    public function getProductsForIndex($storeId, $productIds = null);
}
```

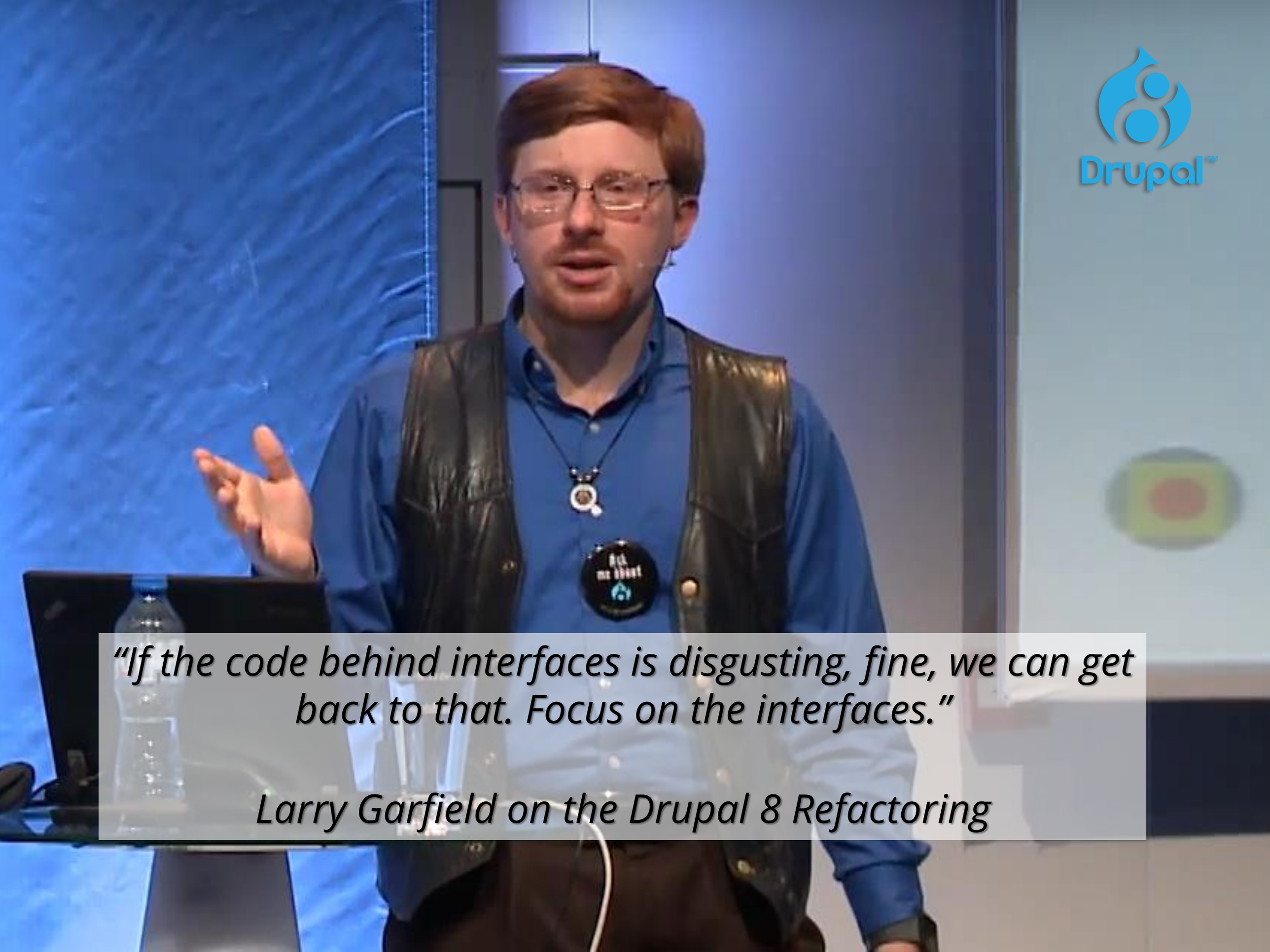
- Useful Design Pattern: Bridge / Adapter
- Module implements library interfaces
  - Product, ProductRepository, …
- Keep interfaces small!
- Interface Segregation
  - HasMediaGallery, HasCategories, …

More on our blog:

https://www.integer-net.com/magento-1-magento-2-shared-code-extensions/

Refactoring: General Tips

- Visualize class dependencies
  - Use doxygen or pen and paper
  - Who is talking to whom?
  - Where are the boundaries?
  - Where do we want them to be?
- Avoid „gold plating“
  - No need for perfect implementation
  - Focus on good abstraction instead
  - Small and well-defined interface is priority



"If the code behind interfaces is disgusting, fine, we can get back to that. Focus on the interfaces."

Larry Garfield on the Drupal 8 Refactoring

More Learnings from Drupal 8

- Don‘t write unit tests for existing stuff
- Deprecate intermediate solutions immediately

„Eating ElePHPants“ Keynote:

https://www.youtube.com/watch?v=5jqY4NNnc3I

Step 2: Port Remaining Module Code

- Tools to get started
  - Unirgy ConvertM1M2: https://github.com/unirgy/convertm1m2
  - Magento Code Migration (official): https://github.com/magento/code-migration
- No stable releases yet
  - Issues with class generation if library code with namespaces is used
- Useful for module XML files

Port Remaining Module Code

- Bottom up, feature by feature
- Drive development with integration tests
- Implement bridge interfaces with unit tests

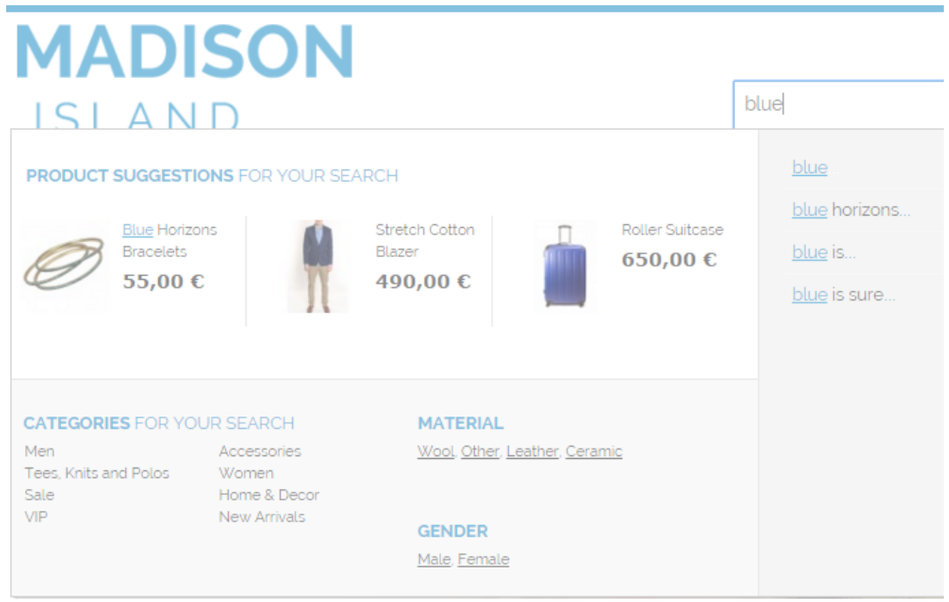
```php
 * @var AttributeRepository
 */
protected $attributeRepository;

/**
 * Constructor
 *
 * @param AttributeRepository $attributeRepository
 * @param StoreManagerInterface $storeManager
 * @param ViewElementContext $context
 * @param array $data
 */
public function __construct(AttributeRepository $attributeRepository, StoreManagerInterface $storeManager,
                            ViewElementContext $context, array $data = [])
{
    $this->attributeRepository = $attributeRepository;
    $this->storeManager = $storeManager;
    parent::__construct($context, $data);
}


public function _toHtml()
{
    $attributes = $this->attributeRepository
        ->getFilterableInSearchAttributes($this->storeManager->getStore()->getId());

    foreach($attributes as $attribute) {
        $this->addOption($attribute->getAttributeCode(), $attribute->getStoreLabel() . ' [' . $attribute->
    }

    return parent::_toHtml();
}

/**
 * @param string $value
 * @return $this
 */
public function setInputName($value)
{
    return $this->setName($value);
}
}
```


```
PHPUnit 4.8.24 by Sebastian Bergmann and contributors.

IntegerNet\Solr\Config
 [x] Acl has access
 [x] Acl no access
 [x] Config section loads

IntegerNet\Solr\Database\AttributeRepository
 [x] It returns filterable in search attributes

IntegerNet\Solr\Module
 [x] The module is registered
 [x] The module is known and enabled
 [x] The module is known and enabled in the real environment
 [x] Dependency injection

IntegerNet\Solr\Model\Bridge\AttributeRepository
 [x] It returns filterable in search attributes
 [x] It uses store id

IntegerNet\Solr\Model\Bridge\Attribute
 [x] Store scope
 [x] Invalid store scope throws exception
 [x] Getter delegation

IntegerNet\Solr\Model\Source\VarcharCategoryAttribute
 [x] It returns filtered category attributes

IntegerNet\Solr\Model\Source\VarcharProductAttribute
 [x] It returns filtered product attributes
```

The Result

https://www.integer-net.com/solr-magento/

- Version 1.5 fully refactored
- Magento 2 Module: work in progress
- Also coming this year: Free version with limited features



More Useful Resources

- SOLID MVC: Framework agnostic domain code
  https://www.youtube.com/watch?v=NdBMQsp_CpE (Stefan Priebsch)

- Mage2Katas: Learn TDD with Magento 2
  http://mage2katas.com/ (Vinai Kopp)

Thank You!

DevelopersParadise

2016 / Opatija / Croatia