MCR 2017

IGOR MELNYKOV

Software Engineer at Magento

@igor_melnikov

Writing extensions best practices

UK.MAGETITANS.COM

#MageTitansMCR @MageTitans

Writing Extensions Best Practices

Igor Melnikov

Software Engineer at
Magento
Austin, TX

Building good quality extensions is hard

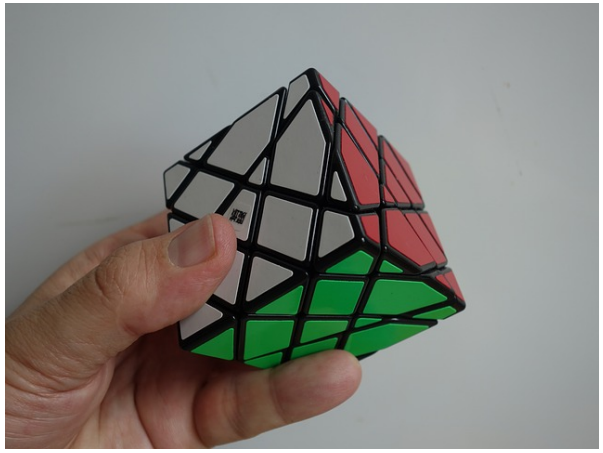
Magento

Some of the Challenges

- Clean and code without bugs
- Compatibility with future Magento releases
- Security
- Conflicts with other extensions
- Extensibility and good API
- Support and improvements after extension is written
  - Backwards compatibility
  - Make sure all functionality works after changes

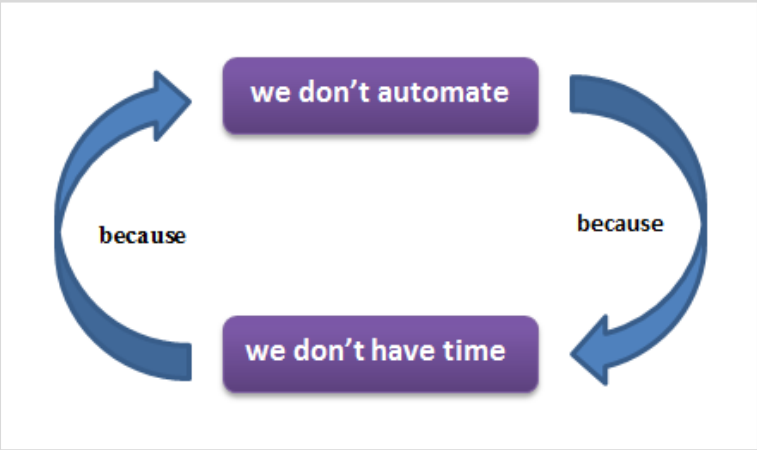
Magento

You can avoid many of the issues if you follow best practices and Magento technical guidelines



Magento

Think about testing and automate critical scenarios



Magento

Value of Test Automation

- Every time you make a change you need to do regression
- Manual testing is too expensive and not reliable
- Unautomated scenarios are your technical debt

Magento

Types of Tests in Magento

Quality

Functional
Integration
Web API
Unit

Most Feedback

Functional
Integration
Web API

Development

Integration
Unit
Static

Magento

Run static tests on your code

Static Tests Help Find …

- Coding style issues
- Coupling between classes and method complexity
- Usage of obsolete classes
- Potential security vulnerabilities
- Undeclared dependencies in composer.json
- References to missing resources
- And more

Magento

How to Run

Using command line

```
$ bin/magento dev:tests:run
```

Manually

```
$ export INCREMENTAL_BUILD=1
$ cd dev/tests/static
$ echo "app/code/MyCompany/MyModule/Model/MyEntity.php" >
    testsuite/Magento/Test/changed_files
$ ../../../vendor/bin/phpunit
```

Set variables_order in php.ini to variables_order = "EGPCS"

Magento

Avoid unnecessary extension conflicts

Magento

What Leads to Extension Conflicts?

- Changing preferences for classes and interfaces
- Around plugins, if the main method is not called
- Rewriting templates
- Rewriting layouts (can and must be avoided)
- Removing blocks from layout
- Depending on specific implementation instead of most generic type

Magento

More on Plugins and Observers

- Create plugins on interfaces
- Declare observers in global area, use specific areas only for presentation layer logic

Magento

Know and follow security best practices

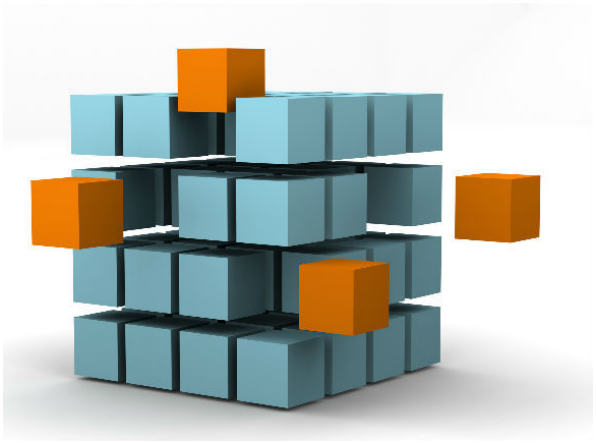
Magento

Security Best Practices

- Be aware how to protect from OWASP Top 10 vulnerabilities
- Use read-only filesystem
- Escape all HTML output (escapeHtml)
- Don’t use native PHP unserialize, use Magento serialize library to secure serialize/unserialize data
- Install latest patches and upgrade Magento

Magento

Follow the modularity



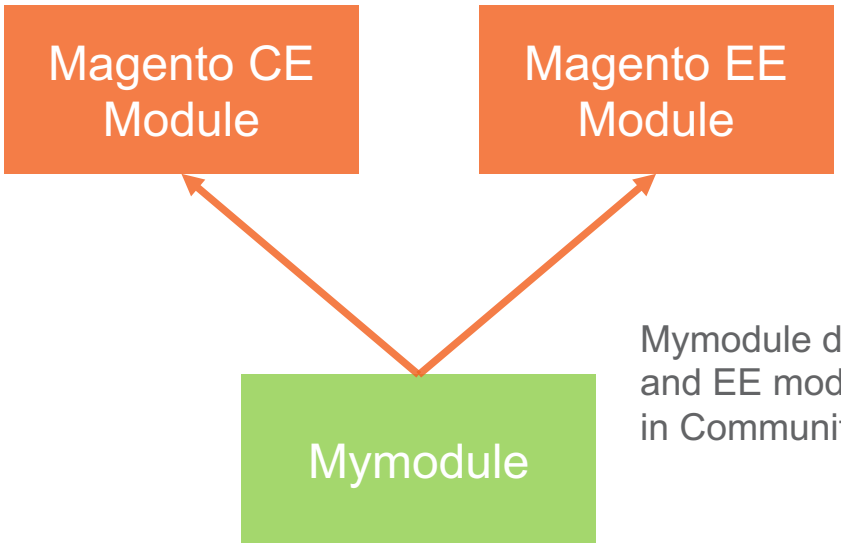
Magento

Modularity

- Declare all dependencies in composer.json
- Reference only **`@api`** code of the modules
- Mark public API and SPI of your module with **`@api`**
- Read more about version dependencies DevDocs/extension-dev-guide/versioning/dependencies.html

Magento

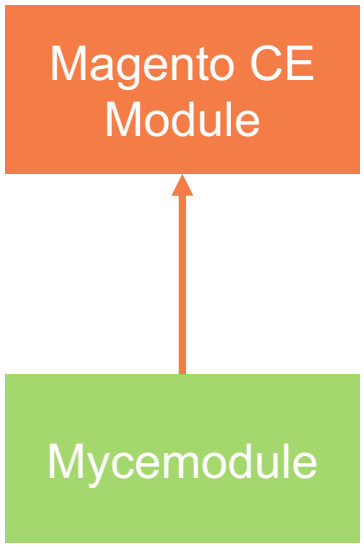
Extension for Multiple Editions



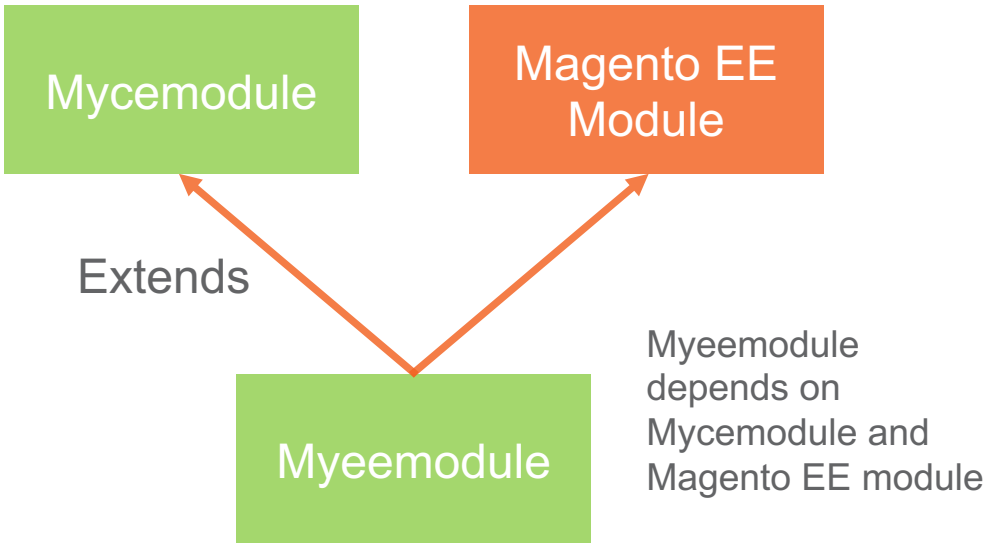
Mymodule depends on Magento CE and EE modules. How to make it work in Community and Enterprise?

Magento

Extension for Multiple Magento Editions



Mycemodule does not depend on EE



Myeemodule depends on Mycemodule and Magento EE module

Magento

Avoid temporal coupling

Magento

Bad Example #1

```
$url = new Url();
$url->setBaseUrl($baseUrl);
echo $url->get('custom/path'); // prints full URL
```

Developer forgot or didn’t know that you need to call setBaseUrl

```
$url = new Url();
echo $url->get('custom/path'); // throws exception
```

Method with out parameters that doesn’t return anything could be sign of temporal coupling

Magento

Good Example #1

```php
$url = new Url($baseUrl);
echo $url->get('custom/path');
```

Or

```php
$url = new Url();
echo $url->get($baseUrl, 'custom/path');
```

Only one way to use API
No temporal coupling

Magento

Bad Example #2

```
class Edit extends Action
{
    public function execute()
    {
        …
        $product = $productResource->load($product, $productSku, 'sku');
        $this->registry->register('product', $product);
    }
}

class View extends Template
{
    public function getProductName()
    {
        $product = $this->registry->get('product');
        return $product->getName();
    }
}
```

Magento

Good Example #2

```
class Edit extends Action
{
     public function execute()
     {
          …
          $product = $productRepository->get($productSku);
     }
}

class View extends Template
{
     public function getProductName()
     {
          …
          $product = $productRepository->get($productSku);
          return $product->getName();
     }
}
```

More flexible
No dependencies between classes
No temporal coupling

Magento

Favor composition over inheritance

Magento

Bad Example

```
class AbstractController
    extends Action
{
    …
    protected function validate(
        $request
    ) {}

    protected function generateHash(
        $request
    ) {}
}
```

```
class Edit extends
AbstractController
{
    public function execute()
    {
        $errors = $this->validate(
            $request
        );
        …
        $hash = $this->generateHash(
            $request
        );
        …
    }
}
```

Magento

Good Example

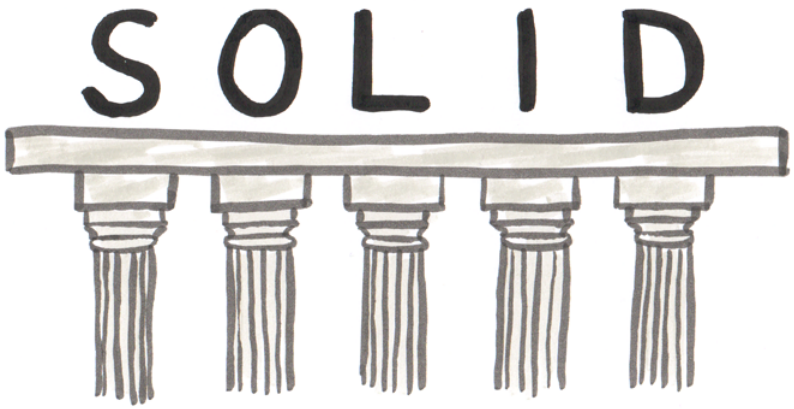
```
class Edit extends Action
{
    public function __constructor(
        ValidatorInterface $validator,
        HashGeneratorInterface $hashGenerator
    ) {}

    public function execute()
    {
        $errors = $this->validator->validate($request);
    }
}
```

- Smaller classes
- One responsibility
- More flexible
- Easy to understand
- More testable

Magento

Follow SOLID principles



Magento

SOLID Principles

- **S**ingle responsibility
- **O**pen closed
- **L**iskov substitution principle
- **I**nterface segregation
- **D**ependency inversion

Magento

Hey, did you just edit the core?
No!
Ok, follow technical guidelines!

Magento

Technical Guidelines

Describes principles and best practices on

- Class design
- Exceptions handling
- Following application layering
- Modularity
- Browser-server interaction
- And more

/Technical Guidelines

Magento

Resources

Resources

Magento technical guidelines
http://devdocs.magento.com/guides/v2.2/coding-standards/technical-guidelines/technical-guidelines.html

Module version dependencies
http://devdocs.magento.com/guides/v2.1/extension-dev-guide/versioning/dependencies.html

SOLID principles
http://www.butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod

Top 10 things which makes your code hard to test
http://misko.hevery.com/2008/07/30/top-10-things-which-make-your-code-hard-to-test/

Effective Java

Pragmatic Programmer

Magento

Q&A

Igor Melnikov
@igor_melnikov

Magento

Thank You