Magento 2

How to write a CRUD Module

Why a CRUD module?

80% of the time, developers write CRUD modules.

Why a CRUD module?

80% of the time, developers write CRUD modules.

Summary

- Module declaration
- Routing
- Menu & ACL
- Controllers
- Models
- Grids
- Blocks
- Templates
- Boilerplate

Resources

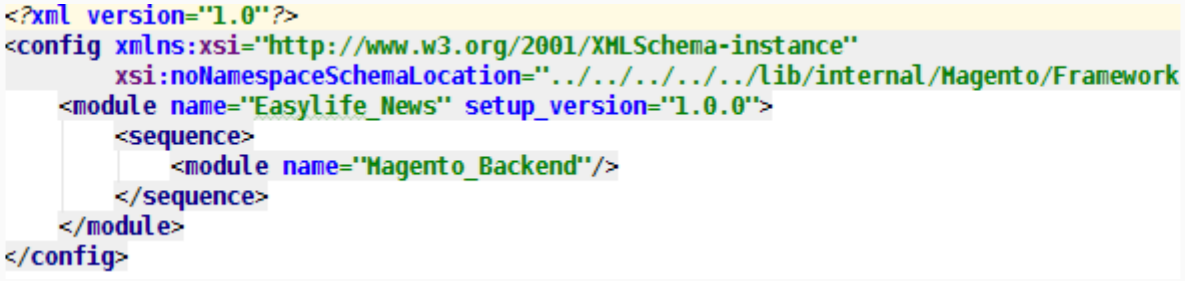
- https://github.com/tzyganu/Magento2SampleModule - Sample CRUD Module for Magento 2.
- https://github.com/ultimatemodulecreator - Ultimate Module Creator for Magento 2.

Differences from Magento 1

- all module files are in the same place. (no more app/code, app/design, skin)
- no more codepools
- no more big fat `config.xml` - instead there are a gazillion small xml files.
- Example: https://github.com/tzyganu/Magento2SampleModule

Where to start?

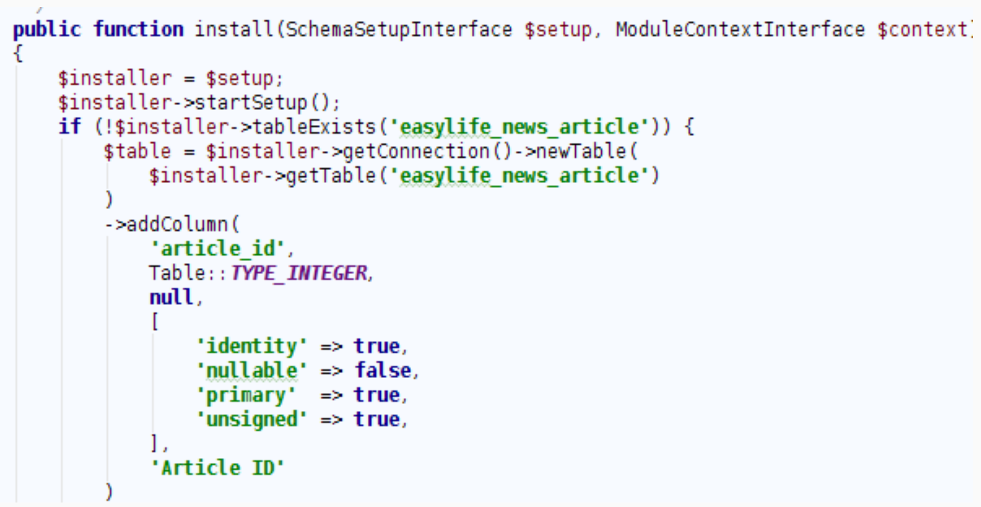
- `app/code/Easylife/News/etc/module.xml`. The equivalent of `app/etc/modules/Easylife_News.xml` from Magento 1.



```xml
<?xml version="1.0"?>
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="../../../../../lib/internal/Magento/Framework
    <module name="Easylife_News" setup_version="1.0.0">
        <sequence>
            <module name="Magento_Backend"/>
        </sequence>
    </module>
</config>
```

Create the installer

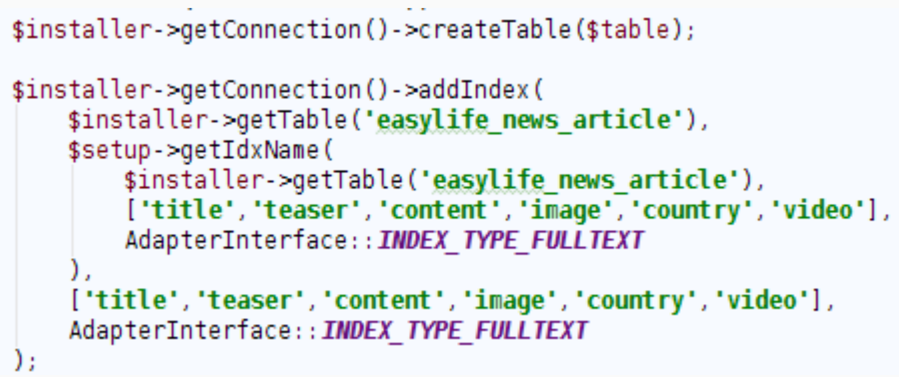
- `Easylife/News/Setup/InstallSchema.php` - Equivalent of `sql/news_setup/install-1.0.0.php`



```
public function install(SchemaSetupInterface $setup, ModuleContextInterface $context
{
    $installer = $setup;
    $installer->startSetup();
    if (!$installer->tableExists('easylife_news_article')) {
        $table = $installer->getConnection()->newTable(
            $installer->getTable('easylife_news_article')
        )
        ->addColumn(
            'article_id',
            Table::TYPE_INTEGER,
            null,
            [
                'identity' => true,
                'nullable' => false,
                'primary'  => true,
                'unsigned' => true,
            ],
            'Article ID'
        )
```

Create the installer

Magento 2 supports mysql5.6+. Full text search can be applied on the Innodb tables also.

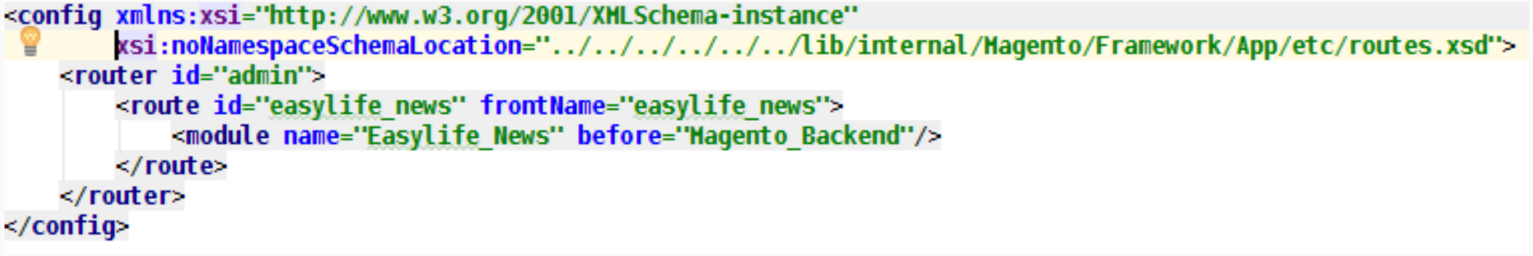


```
$installer->getConnection()->createTable($table);

$installer->getConnection()->addIndex(
    $installer->getTable('easylife_news_article'),
    $setup->getIdxName(
        $installer->getTable('easylife_news_article'),
        ['title','teaser','content','image','country','video'],
        AdapterInterface::INDEX_TYPE_FULLTEXT
    ),
    ['title','teaser','content','image','country','video'],
    AdapterInterface::INDEX_TYPE_FULLTEXT
);
```

Routing

- `etc/adminhtml/routes.xml` for backend
- `etc/frontend/routes.xml` for frontend

in Magento 1 they were stored in `config.xml` under `admin/routers` and `frontend/routers`



```xml
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="../../../../../../lib/internal/Magento/Framework/App/etc/routes.xsd">
    <router id="admin">
        <route id="easylife_news" frontName="easylife_news">
            <module name="Easylife_News" before="Magento_Backend"/>
        </route>
    </router>
</config>
```

Menu & ACL

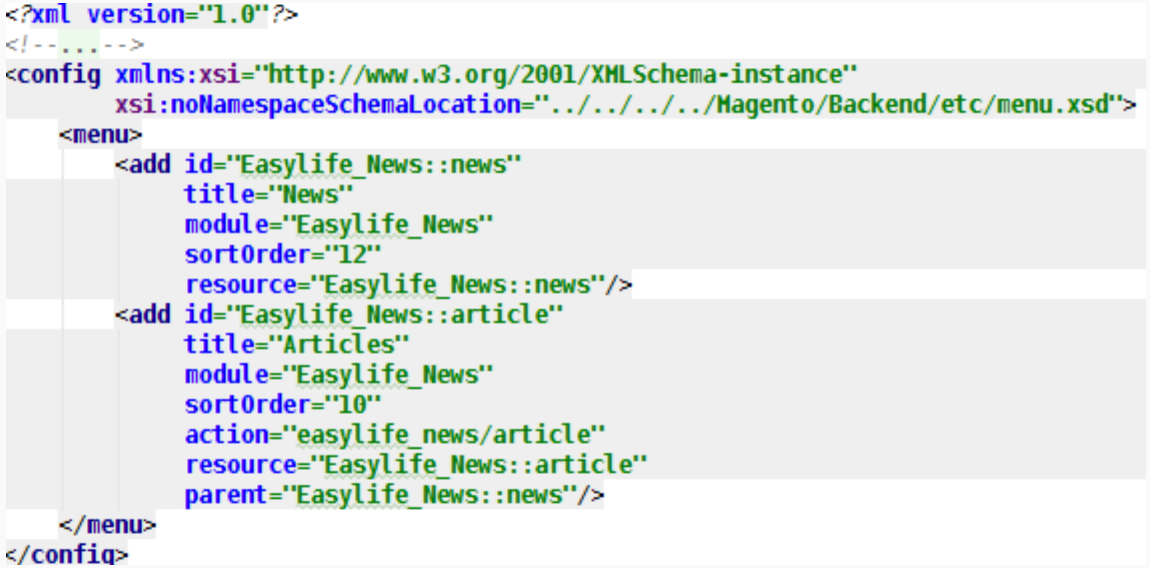
Menu

- moved to `etc/adminhtml/menu.xml` instead of `etc/adminhtml.xml`
- Imperative instead of declarative (add, move, remove directives).

ACL

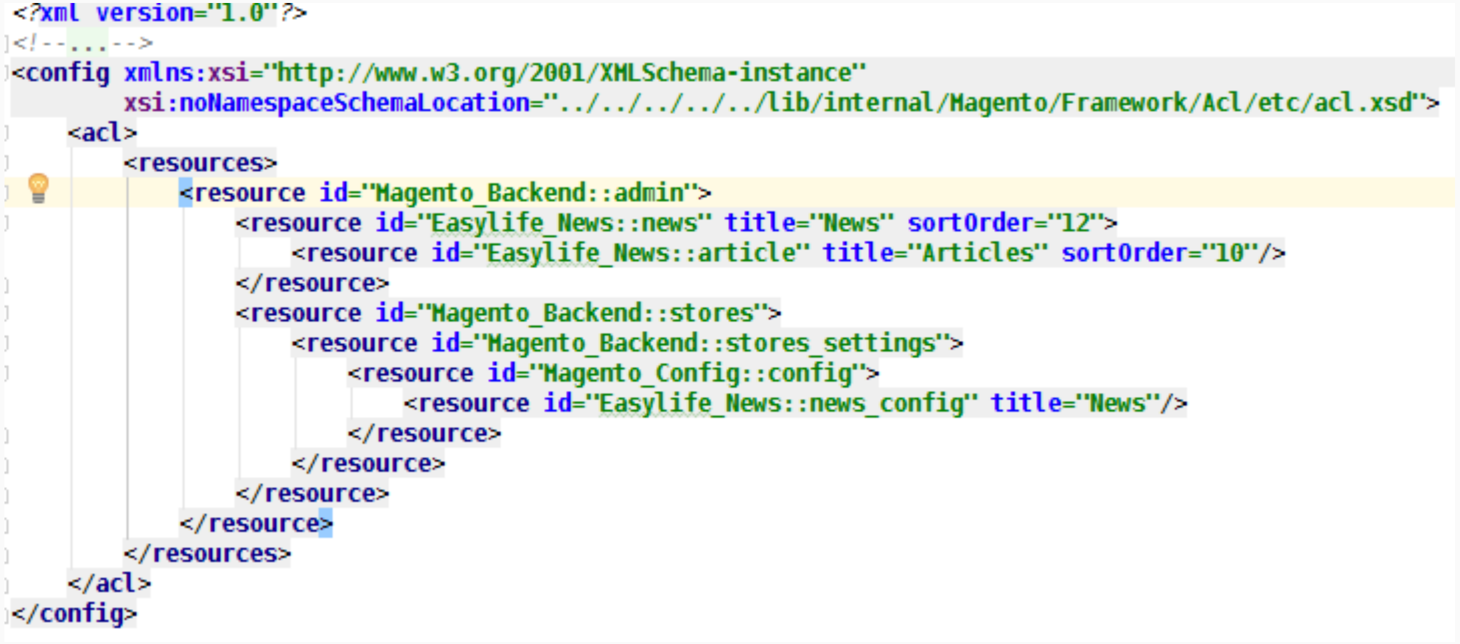
- moved to `etc/acl.xml` instead of `etc/adminhtml.xml`
- Still declarative.

Menu & ACL



```xml
<?xml version="1.0"?>
<!--...-->
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="../../../../Magento/Backend/etc/menu.xsd">
    <menu>
        <add id="Easylife_News::news"
             title="News"
             module="Easylife_News"
             sortOrder="12"
             resource="Easylife_News::news"/>
        <add id="Easylife_News::article"
             title="Articles"
             module="Easylife_News"
             sortOrder="10"
             action="easylife_news/article"
             resource="Easylife_News::article"
             parent="Easylife_News::news"/>
    </menu>
</config>
```

Menu & ACL

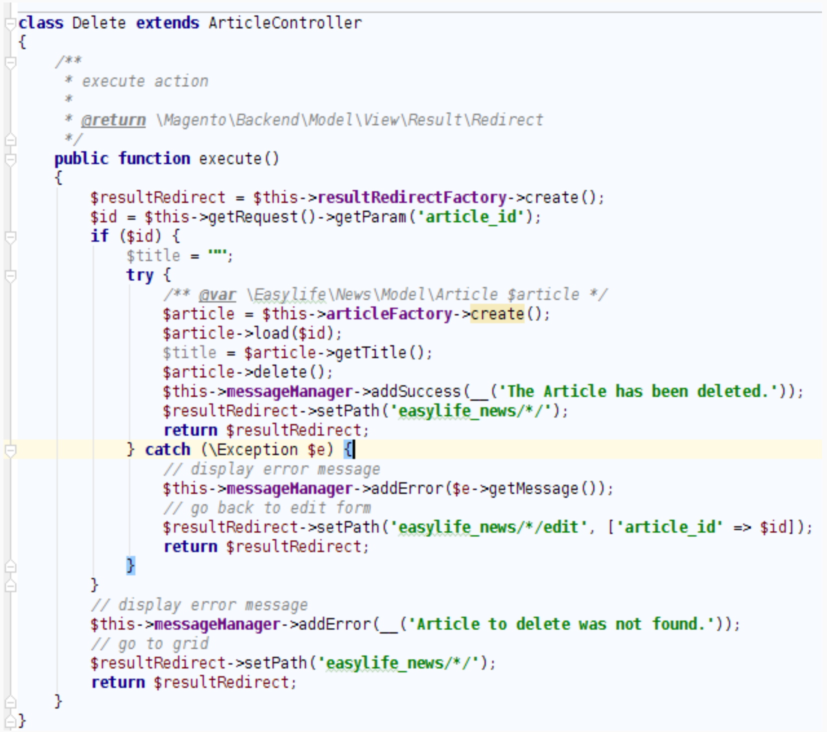


```
<?xml version="1.0"?>
<!--...-->
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="../../../../../lib/internal/Magento/Framework/Acl/etc/acl.xsd">
    <acl>
        <resources>
            <resource id="Magento_Backend::admin">
                <resource id="Easylife_News::news" title="News" sortOrder="12">
                    <resource id="Easylife_News::article" title="Articles" sortOrder="10"/>
                </resource>
                <resource id="Magento_Backend::stores">
                    <resource id="Magento_Backend::stores_settings">
                        <resource id="Magento_Config::config">
                            <resource id="Easylife_News::news_config" title="News"/>
                        </resource>
                    </resource>
                </resource>
            </resource>
        </resources>
    </acl>
</config>
```

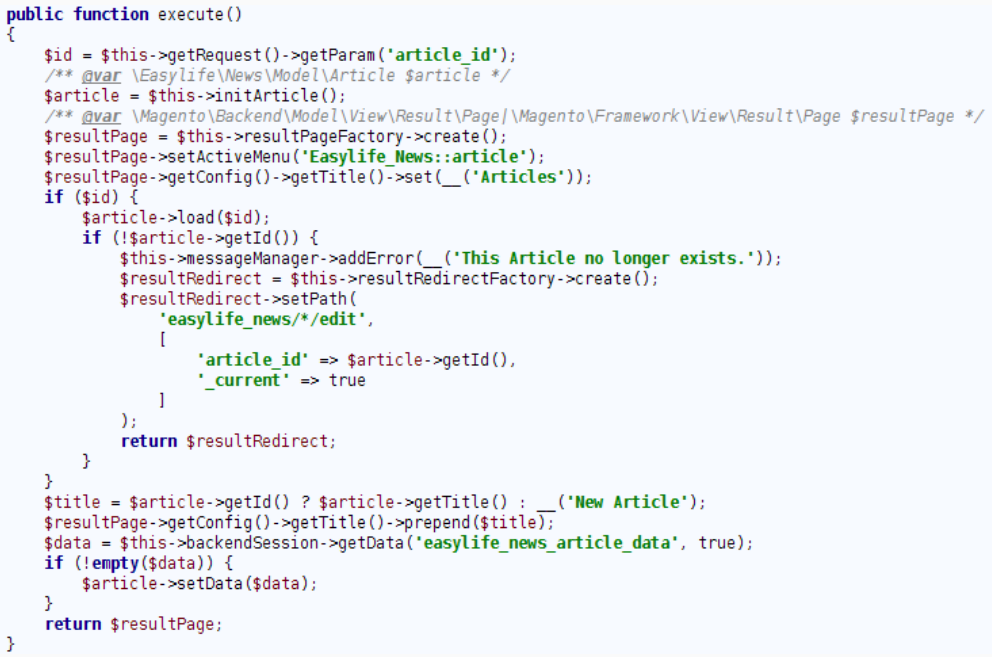
Controllers

- each action is a class and implements the method `execute`
- each action must return an instance of `Magento\Framework\Controller\ResultInterface`
    - `Magento\Framework\Controller\Result\Redirect`
    - `Magento\Framework\Controller\Result\Raw`
    - `Magento\Framework\View\Result\Layout`
    - `Magento\Framework\Controller\Result\Forward`
- `Controller/Adminhtml/Article/Index.php`

Controllers



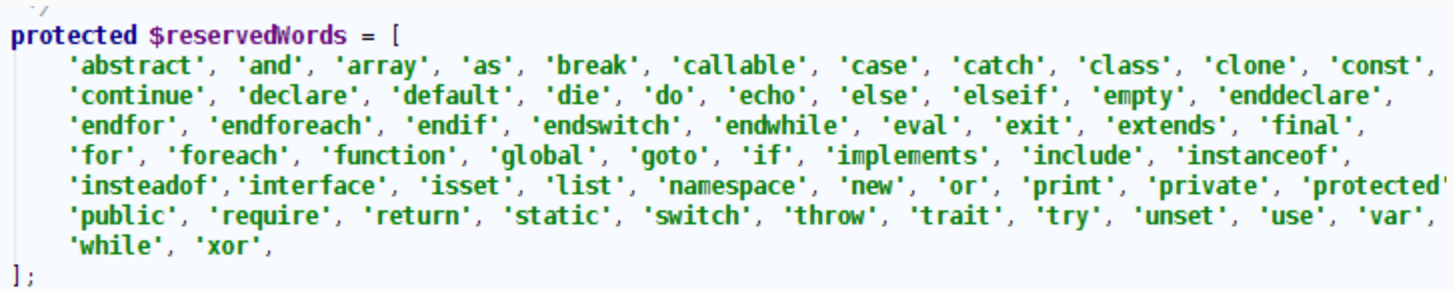
```
class Delete extends ArticleController
{
    /**
     * execute action
     *
     * @return \Magento\Backend\Model\View\Result\Redirect
     */
    public function execute()
    {
        $resultRedirect = $this->resultRedirectFactory->create();
        $id = $this->getRequest()->getParam('article_id');
        if ($id) {
            $title = "";
            try {
                /** @var \Easylife\News\Model\Article $article */
                $article = $this->articleFactory->create();
                $article->load($id);
                $title = $article->getTitle();
                $article->delete();
                $this->messageManager->addSuccess(__('The Article has been deleted.'));
                $resultRedirect->setPath('easylife_news/*/');
                return $resultRedirect;
            } catch (\Exception $e) {
                // display error message
                $this->messageManager->addError($e->getMessage());
                // go back to edit form
                $resultRedirect->setPath('easylife_news/*/edit', ['article_id' => $id]);
                return $resultRedirect;
            }
        }
        // display error message
        $this->messageManager->addError(__('Article to delete was not found.'));
        // go to grid
        $resultRedirect->setPath('easylife_news/*/');
        return $resultRedirect;
    }
}
```



```
public function execute()
{
    $id = $this->getRequest()->getParam('article_id');
    /** @var \Easylife\News\Model\Article $article */
    $article = $this->initArticle();
    /** @var \Magento\Backend\Model\View\Result\Page|\Magento\Framework\View\Result\Page $resultPage */
    $resultPage = $this->resultPageFactory->create();
    $resultPage->setActiveMenu('Easylife_News::article');
    $resultPage->getConfig()->getTitle()->set(__('Articles'));
    if ($id) {
        $article->load($id);
        if (!$article->getId()) {
            $this->messageManager->addError(__('This Article no longer exists.'));
            $resultRedirect = $this->resultRedirectFactory->create();
            $resultRedirect->setPath(
                'easylife_news/*/edit',
                [
                    'article_id' => $article->getId(),
                    '_current' => true
                ]
            );
            return $resultRedirect;
        }
    }
    $title = $article->getId() ? $article->getTitle() : __('New Article');
    $resultPage->getConfig()->getTitle()->prepend($title);
    $data = $this->backendSession->getData('easylife_news_article_data', true);
    if (!empty($data)) {
        $article->setData($data);
    }
    return $resultPage;
}
```

Controllers

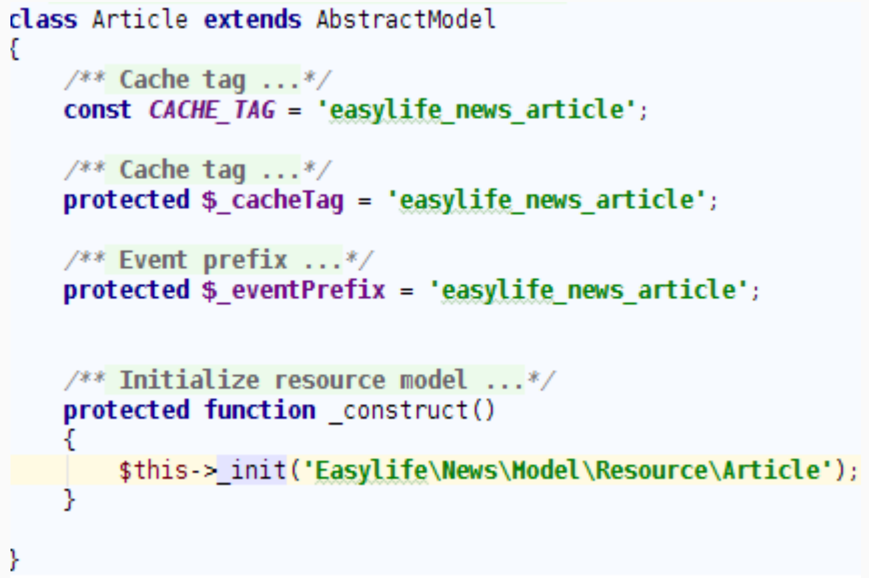
- `Controller/Adminhtml/Article/Edit.php`
- `Controller/Adminhtml/Article/NewAction.php` - What does `Action` do? See `Magento\Framework\App\Router\ActionList`



```
*/
protected $reservedWords = [
    'abstract', 'and', 'array', 'as', 'break', 'callable', 'case', 'catch', 'class', 'clone', 'const',
    'continue', 'declare', 'default', 'die', 'do', 'echo', 'else', 'elseif', 'empty', 'enddeclare',
    'endfor', 'endforeach', 'endif', 'endswitch', 'endwhile', 'eval', 'exit', 'extends', 'final',
    'for', 'foreach', 'function', 'global', 'goto', 'if', 'implements', 'include', 'instanceof',
    'insteadof','interface', 'isset', 'list', 'namespace', 'new', 'or', 'print', 'private', 'protected'
    'public', 'require', 'return', 'static', 'switch', 'throw', 'trait', 'try', 'unset', 'use', 'var',
    'while', 'xor',
];
```

Models

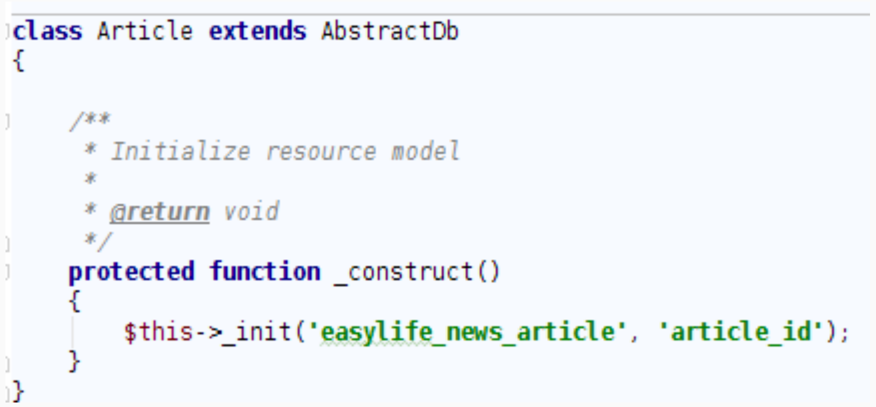
- The entity Model. Similar to Magento 1 module `Easylife\News\Model\Article` extends `Magento\Framework\Model\AbstractModel`



```
class Article extends AbstractModel
{
    /** Cache tag ...*/
    const CACHE_TAG = 'easylife_news_article';

    /** Cache tag ...*/
    protected $_cacheTag = 'easylife_news_article';

    /** Event prefix ...*/
    protected $_eventPrefix = 'easylife_news_article';


    /** Initialize resource model ...*/
    protected function _construct()
    {
        $this->_init('Easylife\News\Model\Resource\Article');
    }

}
```

Models

- The entity resource Model. Similar to Magento 1 module \`Easylife\News\Model\Resource\Article\` extends \`Magento\Framework\Model\Resource\Db\AbstractDb\`

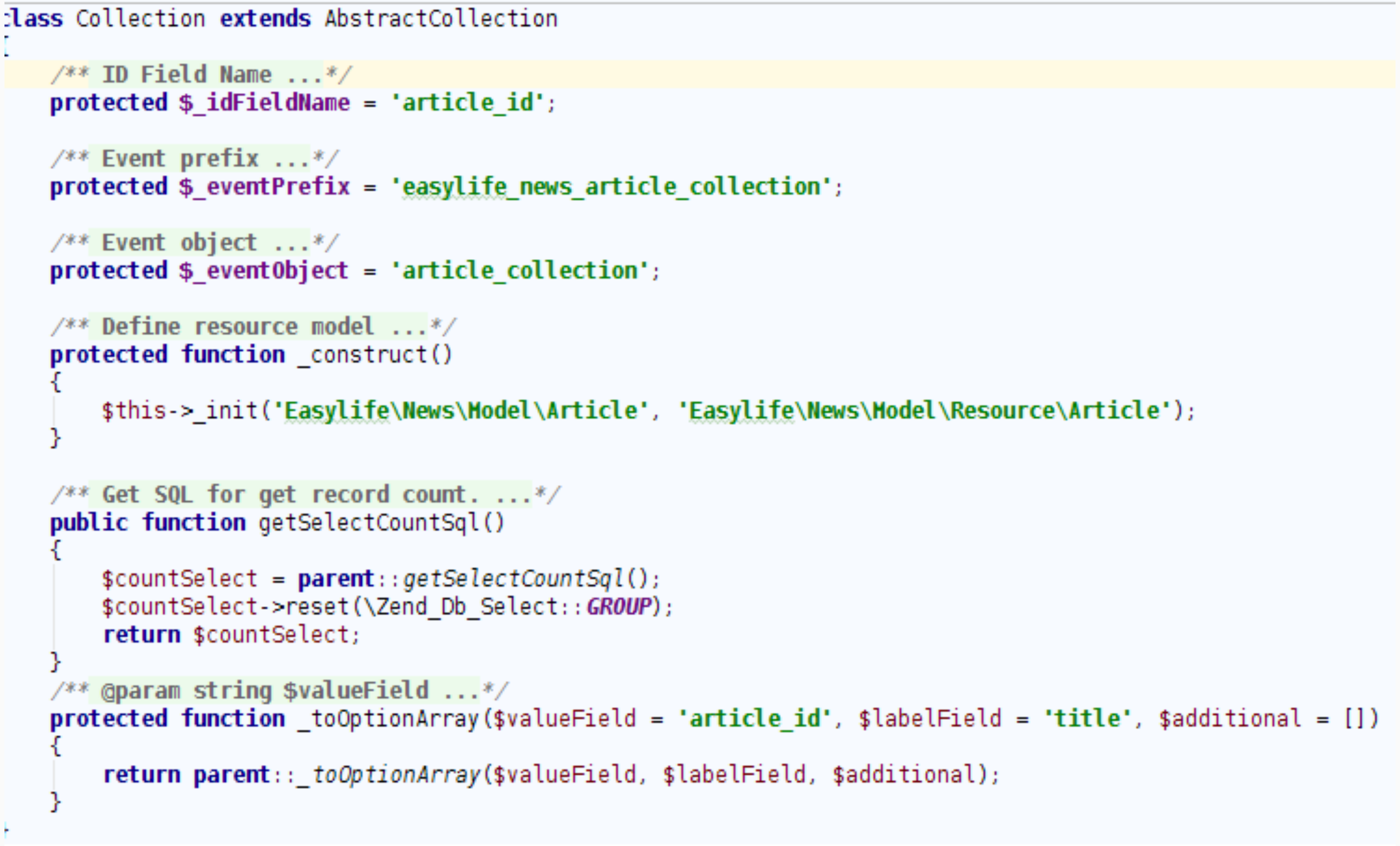


```
class Article extends AbstractDb
{

    /**
     * Initialize resource model
     *
     * @return void
     */
    protected function _construct()
    {
        $this->_init('easylife_news_article', 'article_id');
    }
}
```

Models

- Collection resource model - similar to Magento 1
- `Easylife\News\Model\Resource\Article\Collection` extends `Magento\Framework\Model\Resource\Db\Collection\AbstractCollection`

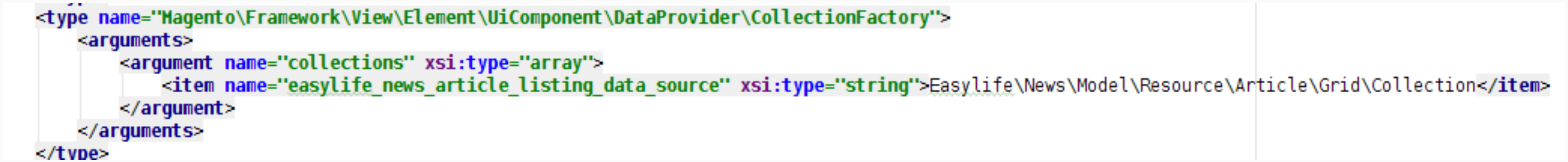
Models



```
lass Collection extends AbstractCollection
{
    /** ID Field Name ...*/
    protected $_idFieldName = 'article_id';

    /** Event prefix ...*/
    protected $_eventPrefix = 'easylife_news_article_collection';

    /** Event object ...*/
    protected $_eventObject = 'article_collection';

    /** Define resource model ...*/
    protected function _construct()
    {
        $this->_init('Easylife\News\Model\Article', 'Easylife\News\Model\Resource\Article');
    }

    /** Get SQL for get record count. ...*/
    public function getSelectCountSql()
    {
        $countSelect = parent::getSelectCountSql();
        $countSelect->reset(\Zend_Db_Select::GROUP);
        return $countSelect;
    }
    /** @param string $valueField ...*/
    protected function _toOptionArray($valueField = 'article_id', $labelField = 'title', $additional = [])
    {
        return parent::_toOptionArray($valueField, $labelField, $additional);
    }
}
```

Models

- The grid collection resource model
- `Easylife\News\Model\Resource\Article\Grid\Collection` extends `Easylife\News\Model\Resource\Article\Collection` implements `Magento\Framework\Api\Search\SearchResultInterface`
- Needed for the new grid system
- Declared in di.xml



```
<type name="Magento\Framework\View\Element\UiComponent\DataProvider\CollectionFactory">
    <arguments>
        <argument name="collections" xsi:type="array">
            <item name="easylife_news_article_listing_data_source" xsi:type="string">Easylife\News\Model\Resource\Article\Grid\Collection</item>
        </argument>
    </arguments>
</type>
```

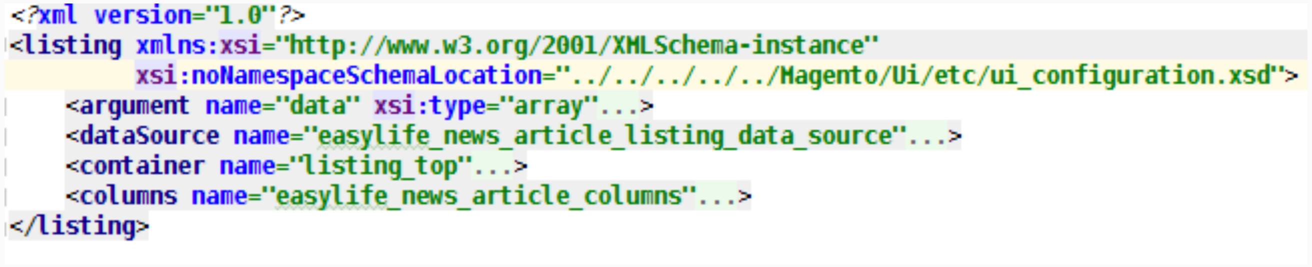
Grids

That’s some crazy but powerful s...tuff in Magento 2.

- Use ui components - a special type of layout
- Knockout.js
- Define grid elements using xml
- full text search
- separate filters from grid rows
- save grid state for later use.

Grids

- `view/adminhtml/ui_component/*.xml`



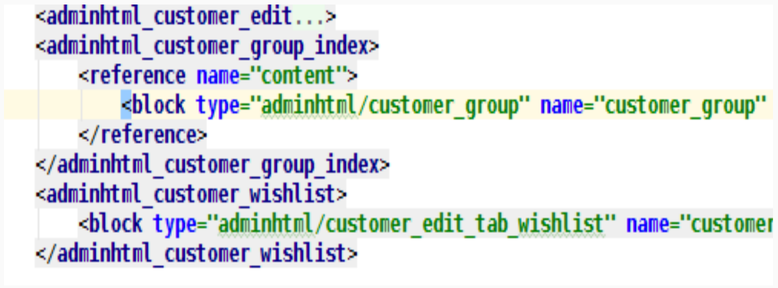
```
<?xml version="1.0"?>
<listing xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="../../../../../Magento/Ui/etc/ui_configuration.xsd">
    <argument name="data" xsi:type="array"...>
    <dataSource name="easylife_news_article_listing_data_source"...>
    <container name="listing_top"...>
    <columns name="easylife_news_article_columns"...>
</listing>
```

From Blocks

- very similar to Magento 1
- `Easylife\News\Block\Adminhtml\Article\Edit`
  - `Easylife\News\Block\Adminhtml\Article\Edit\Form`
  - `Easylife\News\Block\Adminhtml\Article\Edit\Tabs`
    - `Easylife\News\Block\Adminhtml\Article\Edit\Tab\Article`

Layouts

- every layout handle is now a separate file.



```
<adminhtml_customer_edit...>
<adminhtml_customer_group_index>
    <reference name="content">
        <block type="adminhtml/customer_group" name="customer_group"
    </reference>
</adminhtml_customer_group_index>
<adminhtml_customer_wishlist>
    <block type="adminhtml/customer_edit_tab_wishlist" name="customer
</adminhtml_customer_wishlist>
```

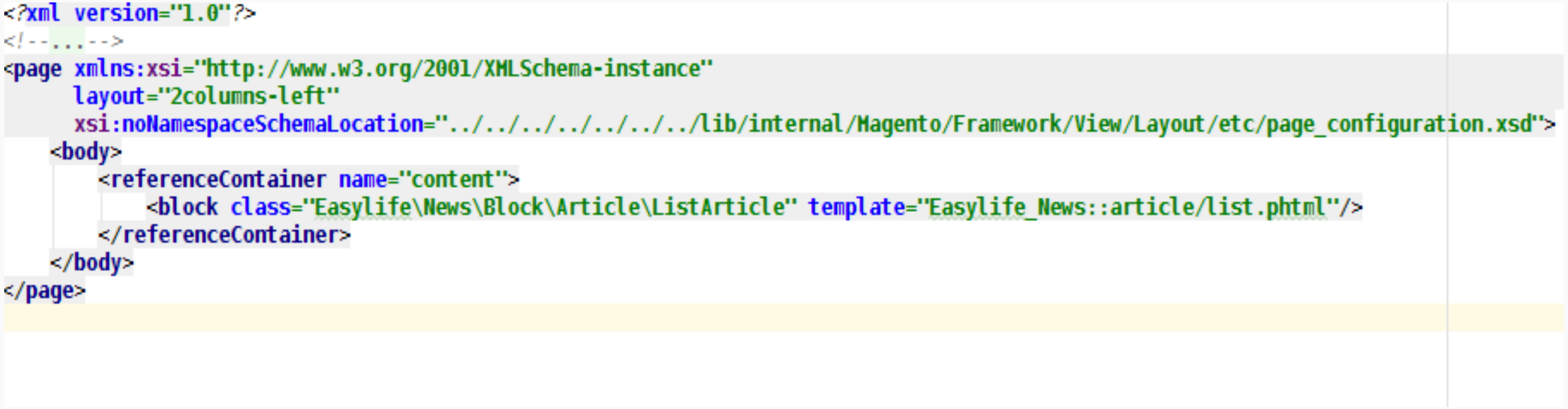
customer_index_edit.xml

customer_group_index.xml

customer_index_viewwishlist.xml

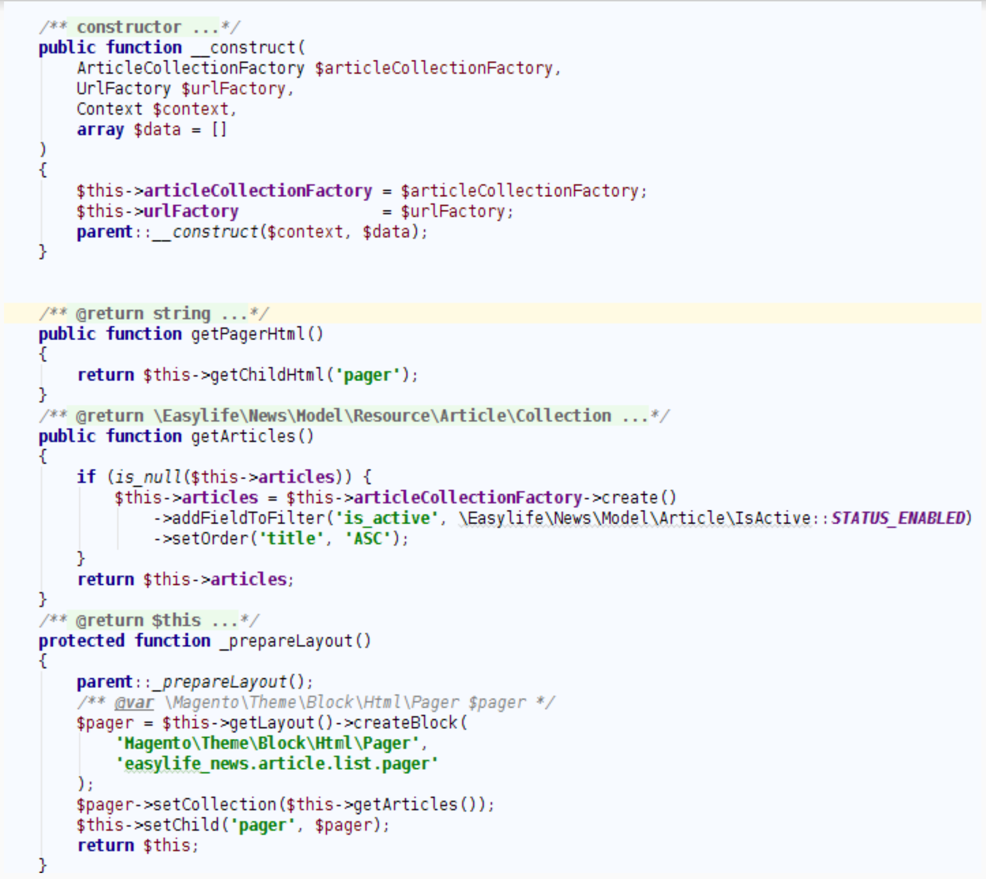
Layouts

- frontend|adminhtml/layout/...



```
<?xml version="1.0"?>
<!--...-->
<page xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      layout="2columns-left"
      xsi:noNamespaceSchemaLocation="../../../../../../../lib/internal/Magento/Framework/View/Layout/etc/page_configuration.xsd">
    <body>
        <referenceContainer name="content">
            <block class="Easylife\News\Block\Article\ListArticle" template="Easylife_News::article/list.phtml"/>
        </referenceContainer>
    </body>
</page>
```

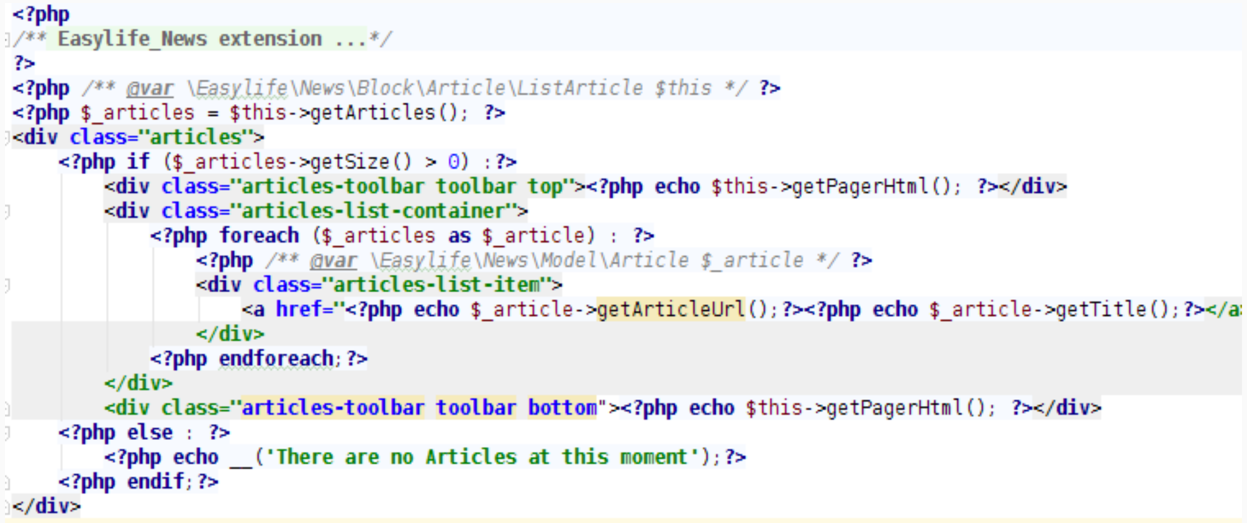
Blocks



```php
/** constructor ...*/
public function __construct(
    ArticleCollectionFactory $articleCollectionFactory,
    UrlFactory $urlFactory,
    Context $context,
    array $data = []
)
{
    $this->articleCollectionFactory = $articleCollectionFactory;
    $this->urlFactory               = $urlFactory;
    parent::__construct($context, $data);
}


/** @return string ...*/
public function getPagerHtml()
{
    return $this->getChildHtml('pager');
}
/** @return \Easylife\News\Model\Resource\Article\Collection ...*/
public function getArticles()
{
    if (is_null($this->articles)) {
        $this->articles = $this->articleCollectionFactory->create()
            ->addFieldToFilter('is_active', \Easylife\News\Model\Article\IsActive::STATUS_ENABLED)
            ->setOrder('title', 'ASC');
    }
    return $this->articles;
}
/** @return $this ...*/
protected function _prepareLayout()
{
    parent::_prepareLayout();
    /** @var \Magento\Theme\Block\Html\Pager $pager */
    $pager = $this->getLayout()->createBlock(
        'Magento\Theme\Block\Html\Pager',
        'easylife_news.article.list.pager'
    );
    $pager->setCollection($this->getArticles());
    $this->setChild('pager', $pager);
    return $this;
}
```

Templates

- frontend/templates/….



```
<?php
/** Easylife_News extension ...*/
?>
<?php /** @var \Easylife\News\Block\Article\ListArticle $this */ ?>
<?php $_articles = $this->getArticles(); ?>
<div class="articles">
    <?php if ($_articles->getSize() > 0) :?>
        <div class="articles-toolbar toolbar top"><?php echo $this->getPagerHtml(); ?></div>
        <div class="articles-list-container">
            <?php foreach ($_articles as $_article) : ?>
                <?php /** @var \Easylife\News\Model\Article $_article */ ?>
                <div class="articles-list-item">
                    <a href="<?php echo $_article->getArticleUrl();?><?php echo $_article->getTitle();?></a
                </div>
            <?php endforeach;?>
        </div>
        <div class="articles-toolbar toolbar bottom"><?php echo $this->getPagerHtml(); ?></div>
    <?php else : ?>
        <?php echo __('There are no Articles at this moment');?>
    <?php endif;?>
</div>
```

Boilerplate

https://github.com/ultimatemodulecreator

✓ Already done:

- Umc_Base - can be used to generate the backend of a module
  - models, resource models, collection resource models
  - adminhtml controllers
  - layout & ui component XMLs
  - Admin form bloks

Boilerplate

❓ Will Follow (don’t know when)

- **Umc_Frontend** - for creating frontend elements: blocks, controllers, layouts, templates, widgets
- **Umc_Catalog** - create a relation between custom entities and catalog entities (products & categories)
- **Umc_Review** - allow users to write reviews for custom entities
- **Umc_Webapi** - generate API for custom entities.
- **Umc_{Whatever}** - waiting for ideas.

Q&A