Design Patterns In Magento

Model View Controller Pattern (MVC)

Design pattern where business, presentation and coupling logic are separated

Front Controller Pattern

Makes sure that there is one and only one point of entry. All requests are investigated, routed to the designated controller and then processed accordingly to the specification

Factory Pattern

Responsible of factorizing (instantiating) classes

Singleton Pattern

Checks the whether this class has already been instantiated before, this results in a shared instance.

Registry Pattern

Internal registry: a global scoped container for storing data

Prototype Pattern

It defines that instances of classes can retrieve a specific other class instance depending on its parent class (the prototype)

Object Pool Pattern

A box with objects so that they do not have to be allocated and destroyed over and over again

Iterator Pattern

The iterator pattern defines that there is a shared way to iterate over a container with objects

Lazy Loading Pattern

Lazy loading ensures that loading data is delayed until the point when it is actually needed

Service Locator Pattern

The service locator pattern abstracts away the retrieval of a certain service. This allows for changing the service without breaking anything

Module Pattern

An implementation of the module pattern would make sure that each element can be removed or swapped

Observer Pattern

By defining observers (or listeners), extra code can be hooked which will be called upon as the observed event fires

Proxy Pattern

Design Patterns In Magento (cont)

👉Provide a surrogate or placeholder for another object to control access to it.
👉Use an extra level of indirection to support distributed, controlled, or intelligent access.
👉Add a wrapper and delegation to protect the real component from undue complexity.

MODULE.XML File

A component declares itself (that is, defines its name and existence) in

```
<?xml version="1.0"?>
<config xmlns: xsi ="ht tp: //w ww.w 3.o rg /20 01/ X
etc/module.xsd">
<module name="V end or_ Com pon ent Nam e" setup_vers
<sequence><module name="Magento_Sales"/></se que nce>
</config>
```

CONFIG.XML File

Used to configure module. The responsibilities of the **config.xml** configu

Example

```
<config xmlns: xsi ="ht tp: //w ww.w 3.o rg /20 01/ X
g.xsd"><default>
<customsetting><general><enable>1</enable><title>Modu
</customsetting></default></config>
```

DI.XML File

What case we use di.xml ?

Example for Preference

```
<p  ref  erence for="Ma  ge n t o  \Cus  to m e r  \A
```

Above code, When someone asks to instantiate a Magento\Customer\A

Example for Arguments

```
<type name="M  ag e n to  \Cu  st o m er  \Mo  de l
c  e\P  ro x y< / arg  ume  nt> </a  rg u m en  ts> <
```


Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

DI.XML File (cont)

In the above code, We are sending object as an argument, we are saying system.xml Proxy class as an object with the name of type Mana

Example for Plugin

```
<type name="M  ag e n to  \Cu  st o m er  \Mo  de l  \Re  sou r ce Mod  el  \V is  ito  r"> <plugin name="c
  \P lu  gin \ Lo g" /> </t  yp e >
```

In the above code , public function clean($object) in visitor class is called after public function afterClean(Visitor $subject, $logResourceModel) whi

Example for Virtual Types

```
<vi  rt u a lType name="o  ur V i rt  ual  Ty p e Na  me" type="P  ul s e st  orm  \T u t or  ial  Vi r t ua
>
```

Creating a virtual type is sort of like creating a sub-class for an existing class. With virtual types, the only behavior you can change in your virtual su

EVENTS.XML File

Custom events can be dispatched by simply passing in a unique event name to the event manager when you call the dispatch function. The unique event name is referenced in your module's **events.xml** file where you specify which observers will react to that event.

The observer xml element has the following properties:

name (required)	The name of the observer for the event definition
instance (required)	The fully qualified class name of the observer
disabled	Determines whether this observer is acti
shared	Etermines the lifestyle of the class. Default is false

Example

```
<event name="my_module_event_before">
<observer name="m  yO b s er  ver  Na m e " instan  ce
\My  Ob s e rv  er" />
</event>
```

ACL.XML File

file is where we define our module access control list resources. Access control list resources are visible under the Magento admin System | Permi

Example

```
<?xml version="1.0"?>
<config xmlns: xsi ="ht tp: //w ww.w 3.o rg /20 01/ XML
<acl><resources><resource id="Magento_Backend::admin">
<resource id=" Max ime _Jo bs: :jo b_h ead " title= " Job s" sortOr der ="10 0" >
" Delete Depart men t" sortOr der ="20 " /></resource></resource>
</resource></resources></acl>
</config>
```

SYSTEM.XML File

The system.xml file is a configuration file which is used to create config
uration fields in Magento 2 System Configuration. You will need this
if your module has some settings which the admin needs to set

MENU.XML File

A **menu.xml** file is a collection of <add/> nodes. Each of these nodes ac

Example

```
<config xmlns: xsi ="ht tp: //w ww.w 3.o rg /20 01/ X
<menu><add id=" Exa mpl e_M enu Tut ori al: :my _me n
t o_B ack end ::c ont ent " /></menu>
</config>
```

id	Attribute defines a unique identifier for this node. E
title	Attribute controls the text an end users sees for th
module	Attribute should match the current module
sortOrder	Attribute controls how this Menu Item is sorted witl
resource	attribute defines the ACL rule a user must have in

WIDGET.XML File

The file that is responsible for widgets in Magento is placed in **[module_dir]/etc/widget.xml**

WEBAPI.XML File

The **webapi.xml** is used to configure access rights and API Interface wh

Example

```
<?xml version="1.0"?>
<routes xmlns: xsi ="ht tp: //w ww.w 3.o rg /20 01/ X
etc/webapi.xsd">
<route url="/V 1/h ell o/n ame /:n ame " method="GET"
<service class= " Inc hoo \He llo \Ap i\H ell oIn te
<resource ref="anonymous" />
</resources></route></routes>
```

Resource tag defines what resources user needs to have to be able to o_Customer::group

Magento 2 Modes



Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

>_ Command line

Cache Management	Commands
Clear all caches:	php magento cache:clean
Clearing a single cache:	php magento cache:clean config
Flush all caches:	php magento cache:flush
Flushing a single cache:	php magento cache:flush config
Index Management	**Commands**
Runs all available indexes:	php magento indexer:r-eindex
Cron	**Commands**
Runs all cron jobs by schedule:	php magento cron:run
Help & Knowledge	**Commands**
Display help/infos about a command	php magento help indexer:r-eindex
Lists all available CLI commands	php magento list
Developement & Production	**Commands**
Set to the production mode to harden your setup	php magento deploy:mode:-setproduction
(exceptions are not displayed to the user, only logged to log files)	
Developer mode is less secure than Production mode	php magento deploy:mode:-setdeveloper
provides verbose logging (errors displayed in browser) enhanced debuggingand disables static view file caching	
Recompile Store	**Commands**
Recompile the Magento 2 codebase	php magento setup:di:-compile

♻ Useful Code Snippets and Tips

How to get product collection in custom template block

```
$objec tMa nager = \Magen to \Fra mew ork \Ap p\O bje ctM ana ger ->g etI nst ance();
 /** @var \Magen to \Cat alo g\M ode l\R eso urc eMo del \Pr odu ct \Col lection $produ ctC oll ection *
/
$productCollection = $objec tMa nag er- >cr eat e(' Mag ent o\C ata log \Mo del \Re sou rce Mod el \Pro -
duc t\C oll ect ion');
/** Apply filters here */
$productCollection->load();
```

How to use Mage::log method in Magento 2

```
protected $logger;
public function __cons tru ct( \Ps r\L og \Log ger Int erface $logger) {$this ->l ogger = $logger; }
```

♻ Useful Code Snippets and Tips (cont)

You use debug, exception, system for psr logger for example

```
$this- >lo gge r-> inf o($ mes sage);
$this->logger->debug($message);
```

How to retrieve product information in Magento 2

In Magento 2 proposed to use service layer for this. Try use `\Magen t`

How to get a store in Magento 2 programmatically?

Magento 1.x

```
Mage:: app ()- >ge tSt ore()
```

```
function rm_sto re($id = null) {
/** @var \Magen to \Fra mew ork \Ob jec tMa nag erI r
$om = \Magen to \Fra mew ork \Ap p\O bje ctM ana ger
/** @var \Magen to \Sto re \Mod el \Sto reM ana ger
$manager = $om->get('Magento\Store\Model\StoreManage
return $manag er- >ge tSt ore ($id); }
```

How to make JavaScript strings translatable (localizable)

To make JavaScript strings translatable or localizable in Magento 2, us

Example : $.mage.__('Select type of option.')

Some Magento 2 core modules use another way to translate JavaScrip

How to call theme images from a static block

You call theme images from static block as follows: **{{view url="images/**

How to instantiate a model in Magento 2

Magento strictly discourages the use of ObjectManager directly. It provi
For all non-injectables (models) you have to use factory:

```
protected $pageFactory;
public function __cons tru ct( \Ma gen to \Cms \Mo de
$this->pageFactory = $pageF actory;
}
public function someFunc() {
... $page = $this- >pa geF act ory ->c rea te(); ...
}
```

How to get the collection of custom Magento 2 modules

```
protected $mymod ule mod elF actory;
public function __cons truct( .... \[Name spa ce] \[
od elF actory,  ... ) {
 ... $this- >my mod ule mod elF actory = $mymod ule
}
```

and you can use in any one of the class methods:

```
$collection = $this->mymodulemodelFactory->create();
```

How to get POST and GET requests in Magento 2

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

♻ Useful Code Snippets and Tips (cont)

In a case of a controller that extends **Magento\Framework\App\Action\Action**, it is possible to get the request with the aid of `$this- >ge tRe que st( )-> get Post().`

For a custom class, inject the request in the constructor:

```
namespace Namesp ace \Mo dul e\S ome thing;
class ClassName  {
protected $request;
public function __cons truct(
\Magento\Framework\App\Request\Http $request,
 ....//rest of parameters here     ) {
$this->request = $request;
...//rest of constr uctor here
}
public function getPost()     {
return $this- >re que st- >ge tPo st();
}
}
```



By **Abdellatif EL MIZEB**
(latmiz)
cheatography.com/latmiz/

Published 21st February, 2017.
Last updated 21st February, 2017.

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com