

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

EVENTS.XML FILE

Custom events can be dispatched by simply passing the above code. We are sending an event using the dispatch function. Your unique event name is replacing the existing `dispatch` events.xml file 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 active. Default value is false
shared	Determines the lifestyle of the class. Default is false

Example

```
<event name="my_module_event_before">
<observer name="myObserverName"
instance="MyObserver" />
</event>
```

DI.XML File

What case we use di.xml ?

Example for Preference

```
<preference for="Magento\Customer\Api\AddressRepositoryInterface"
to="Magento\Customer\Api\AddressRepository" />
```

Above code, When someone asks you to instantiate a `Magento\Customer\Api\AddressRepository` class.

Example for Arguments

```
<type name="Magento\Customer\Model\Group"
arguments="Magento\Customer\Model\Group" />
```

DI.XML File (cont)

In the above code, we are sending an event using the dispatch function. Your unique event name is replacing the existing `dispatch` events.xml file which observers will react to that event.

Example for Plugin

```
<type name="Magento\Customer\Model\Group"
arguments="Magento\Customer\Model\Group" />
```

In the above code, public function `clean($object)`

Example for Virtual Types

Creating a virtual type is sort of like creating a class. It is a class that is not actually compiled into the system.

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.

Above code, When someone asks you to instantiate a `Magento\Customer\Api\AddressRepository` class.

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.

Design Patterns In Magento (cont)

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.



By [deleted]
cheatography.com/deleted-34559/

Not published yet.
Last updated 15th February, 2017.
Page 2 of 2.

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