Magento 2

1. **Difference between Magento1 and Magento2?**

|  |  |
| --- | --- |
| **Magento 1.0+** | **Magento 2.0+** |
| API Added Retroactively | API is Core to the technology |
| Flexible Architecture | New Architecture for Speed |
| Open Source (CE) | Open Source (CE) |
| No Front-end Library | Ships with LESS |
| Legacy PHP | PHP5.6+ / 7.0 |
| No Support for HTML5/CSS3 | Native Support HTML5/CSS3 |
| RWD Theme Included | RWD Theme Included |
| Weak Content Staging | Advanced Content Staging |
| Sub-Par Search (SOLR) | Elastic Search (EE) |
| External PayPal | In-Site PayPal Experience |
| Secure Bridge | PCI Compliance Hosted Fields |
| Minimal Video Features | Integrated Video (PDP)\* |
| Full Page Cache (EE) | FPC (EE)/Varnish |
| Severe DB Locking Issues | Zero Table Locking (Checkout) |
| Decent Automated Tests | Better Automated Tests |
| JS – Prototype | JS – Jquery |
| Huge Extension Marketplace | Minimal Extensions for 2.0 |
| Stale Admin Panel (Backoffice) | Fresh New Design in Admin |
| No Data Grid | Customizable Data Grid for Catalog |
| Admin Non-Responsive | Responsive Admin Panel |
| EE Pricing by License | EE Pricing by Order Volume |
| 5-12 Month Project Cycle | Claimed 4-8 Month Project Cycle\* |
| Legacy Checkout | Streamlined Checkout Process |
| Popular Payment & Shipping Methods | Same Payment & Shipping Methods |
| “Wild West” Extension Market | Rigorous Extension Testing |
| Connect Store | New “Marketplace” |

Notable Architecture differences are:

1. Magento 2 supports latest PHP versions. These versions include security improvements that affect store’s speed.
2. Reduction in unnecessary browser’s operations on the client’s side due to bundled and minimized JavaScript.
3. Better browser caching for static content.
4. Speed & Performance: Magento 2 offers faster page load speed when compared to Magento 1. Thanks to full-page caching Magento has offered in both Community as well as Enterprise Edition.

|  |  |
| --- | --- |
| **Magento 1** | **Magento 2** |
| Apache 2.x | Apache 2.2 / 2.4 |
|  | Nginx 1.7 or greater |
| PHP 5.2.x – 5.5.x | PHP 5.6.x / 7.0.2 / 7.0.6 |
| MySQL | MySQL / MySQL Percona 5.6.x or greater |
|  | Varnish 3.x / 4.x |
|  | Redis 2.x / 3.x or Memcached 1.4.x |
| Solr (Only for EE) | Solr 4.x |
| HTML | HTML5 |
| CSS | CSS3 |
| JQuery (In latest themes) | JQuery |
|  | RequireJS / Knockout.js |
| Zend Framework 1 | Zend Framework 1 / 2 |
|  | Symfony |
|  | Composer |
|  | PSR – 0 / 1 / 2 / 3 / 4 |

1. **Types of plugins?**

In Magento2, we can create and use three types of Plugin.

* 1. Before listener Plugin
  2. After listener Plugin
  3. Around listener Plugin

***A. Before Listener***

Before listeners are used whenever we want to change the arguments of an original method or want to add some behavior before an original method is called.

We will use before listeners to change behavior of addProduct method of Magento\Checkout\Model\Cart.

***B. After Listener***

After listeners are used when we want to change values returned by an original method or want to add some behavior after an original method is called.

We will use after listeners to change behavior of getName method of Magento\Catalog\Model\Product.

***C. Around Listener***

Around listeners are used when we want to change both the arguments and the returned values of an original method or add some behavior before and after an original method is called.

We will use around listeners to change behavior of addProduct method of Magento\Checkout\Model\Cart.

Around listener is called by adding prefix ‘arround’ to the method name and setting first letter of original method to capital.

Now method addProduct will become aroundAddProduct.

The around listener methods must have a return value.

The return value is formed in such way that the parameters following the $closure parameter in the around listener method definition are passed to the $closure function call in a sequential order.

1. **How to Override Model in Magento 2?**

## Steps to Follow

* Create Directories
* Configuration of Module
* Registration of Module
* Override di.xml
* Override Product.php
* Launch SSH and Run Commands

1. **magento 2 override protected function?**

You can not override a protected function. However you can override the public method which is calling that protected method. In my case I needed to override the method called insertLogo. However this being the protected method I could not override. So I override the Invoice.php which was calling insertLogo in getPdf method. In the same file I redefined the code for insertLogo.

1. **How to override core function?**
2. **What is the URL pattern in Magento?**
3. **What is the Magento flow when you hit the url?**

**Step 1 :** Request hits index.php in root folder of Magento 2.

**Step 2 :** index.php calls static create() method of bootstrap class to create objectmanagerfactory and bootstrap objects.

**Step 3 :** index.php calls createApplication() method of bootstrap object which in turn will initiate ObjectManager using objectManagerFactory. Then, application instance gets created using objectManager and returned to index.php.

**Step 4 :** index.php calls run() method of bootstrap object to run the application, which in turn will call launch() method of application object.

index.php creates application instance (\Magento\Framework\App\Http) using objectManager (that is created before that in bootstrap) and calls method launch.

**Step 5 :** HTTP Application instance does initial routing. It takes first part of the URL to find out which area should be loaded (admin/frontend).

When the area is defined, configuration for that area is loaded (unlike Magento 1 where all configuration was loaded).

Then application object calls Magento\Framework\App\FrontControllerInterface::dispatch of requested area.

**Step 6 :** Front Controller does routing same as in Magento 1. All routers are iterated to match request. The router that matches the request returns instance of ActionController. When FrontController gets instance of action controller it calls dispatch method on it. This process is more efficient because only routers of one area are used (admin router is not used in frontend).

**Step 7 :** Action controller performs its task and returns instance of some implementation of ResultInterface.

**Step 8 :** FrontController just returns ResultInterface to Application Instance that renders the response.

1. **Cronjob setup in Magneto**
2. **In order to create a custom cron job, please follow the below steps.**
   1. Step 1: Create a sample module.
   2. Step 2: Create a class to run cron.
   3. Step 3: Create crontab.xml.
   4. Step 4: Run the cron job.
   5. Step 5: Create custom cron group.
   6. Step 6 : Run the custom cron group.
3. **Types of deploy modes?**

There are three primary Magento 2 modes:

* 1. Default Mode
  2. Developer Mode
  3. Production Mode

There is also another mode known as “Maintenance Mode,” but it operates very differently by preventing access to the system.

***Default Mode***

As the name describes, the Default mode is how Magento 2 works, if no other mode is specified. It lets you deploy Magento on a single server without changing any settings and configurations. However, the Default mode is not optimized for production.

The Default mode has the following key features:

* Errors and Exceptions are never shown to the user. They are logged to var/reports directory at the server.
* Symlinks to the static view files are published to the pub/static.
* The Default mode hides custom X-Magento-\* HTTP request and response headers
* This mode is not optimized for the production environment because the static view files are first materialized and then cached.

***Developer Mode***

This mode is recommended for developers and is the recommended mode for the development. You should set the Magento 2 Developer mode when you need to customize and extend the platform’s functionalities. This mode is slower on the front end and the website admin.

Some other features of the Developer mode include:

* Uncaught exceptions and errors displayed in the web browser.
* Enhanced debugging.
* Verbose system logging.
* The exception is thrown in the web browser when an event subscriber cannot be invoked.
* The exception appears in the error handler rather than being logged.
* Automatic code compilation.

***Production Mode***

When your store is ready to go and has been deployed to a production server, you should set the Magento 2 in production mode. After that, you should also deploy the static view files to the Magento 2 pub/static directory because this improves the performance of the store.

In production mode:

* Static view files are not materialized. They are only served from the cache.
* Errors are never displayed to the user. Instead, they are logged to the filesystem.

1. **What is DI**
2. **What is factory**
3. **What is object manager**
4. **How to get particular record (i.e. column\_name=”test”) from Collection?**
5. **How to create custom module?**
6. **How to do database operations in module?**
7. **CI Mangeto commands**
8. **Difference between cache:clean and cache:flush**
9. **How to show different prices for single customer?**
10. **What is UIComponent in admin?**
11. Where to change (filename) the database name in Magento?
12. Realtime example of Website, Store and StoreView

2nd Round:

* + - 1. What you did in your project? And tell the concept you used to achieve it?
      2. How to show discount for bulk order by single/multiple customer?
      3. Difference between plugins and event observers?
      4. Difference between cache:clean and cache:flush