EBook Gratuito

APPRENDIMENTO magento2

Free unaffiliated eBook created from
Stack Overflow contributors.

#magento2

Sommario

Di

Capitolo 1: Iniziare con Magento2

Osservazioni

Magento 2 è una piattaforma di e-commerce open source progettata per facilitare la struttura comune del carrello degli acquisti per le pagine Web. Rispetto alle precedenti versioni di Magento, la versione 2.0 è più snella e performante, eliminando i problemi con il blocco delle tabelle e migliorando il sistema di pagamento per gli utenti ospiti.

Versioni

Versione	Data di rilascio
2.1.7	2017/05/31
2.1.6	2017/04/11
2.1.5	2017/02/21
2.1.4	2017/02/07
2.1.3	2016/12/14
2.1.2	2016/10/10
2.1.1	2016/08/25
2.1.0	2016/06/23
2.0.14	2017/05/31
2.0.13	2017/02/21
2.0.12	2017/02/07
2.0.11	2016/10/12
2.0.10	2016/10/07
2.0.9	2016/08/04
2.0.8	2016/07/18
2.0.7	2016/05/19
2.0.6	2016/05/13
2.0.5	2016/04/27

Versione	Data di rilascio
2.0.4	2016/03/31
2.0.3	2016/03/30
2.0.2	2016/01/28
2.0.1	2016/01/19
2.0.0	2015/11/17

Examples

Installazione o configurazione

Istruzioni dettagliate su come installare o installare Magento2.

Installa Magento 2 su Ubuntu 16.04

NOTE: Installeremo Magento 2 su Ubuntu Server 16.04 LTS nuovo con PHP 7.0, MySQL 5.6 e Apache 2.4.

1. Requisiti di installazione

- Apache 2.2 o 2.4 con modulo mod_rewrite (o) Nginx> = 1.8.
- PHP 5.5 o versione successiva. PHP 7.0 anche supportato.
- Moduli PHP richiesti: PDO / MySQL, mbstring, mcrypt, mshsh, SimpleXML, curl, xsl, gd, ImageMagick 6.3.7 (o successivi) o entrambi, soap, intl, openssl.
- Compositore e Git.

È possibile utilizzare il seguente comando per installare tutti i requisiti di cui sopra dal repository predefinito (xenial).

```
sudo apt install apache2 git mysql-server
sudo apt install php libapache2-mod-php php-mysql php-dom php-simplexml php-gd
sudo apt install php-curl php-intl php-xsl php-mbstring php-zip php-xml php-mcrypt
```

Raccomando di installare dalla homepage invece del repository di Ubuntu.

```
curl -sS https://getcomposer.org/installer | php
mv composer.phar /usr/local/bin/composer
chmod +x /usr/local/bin/composer
```

2. Imposta Magento 2

a) Scarica da GitHub

Il codice Magento2 è disponibile sotto il repository Github. Usa il seguente comando per clonare il repository Magento2 sul tuo sistema.

```
cd /var/www/
git clone https://github.com/magento/magento2.git
```

b) Download tramite Composer

Se non vuoi installare Magento 2 clonando da GitHub, va bene. Puoi anche installarlo tramite Composer.

```
cd /var/www
composer create-project --repository-url=https://repo.magento.com/ magento/project-community-
edition magento2
```

Ora installa tutti i moduli necessari per Magento2 usando il compositore. Attendere il completamento del processo di installazione. (Non ti servirà se stai installando Magento 2 tramite Composer)

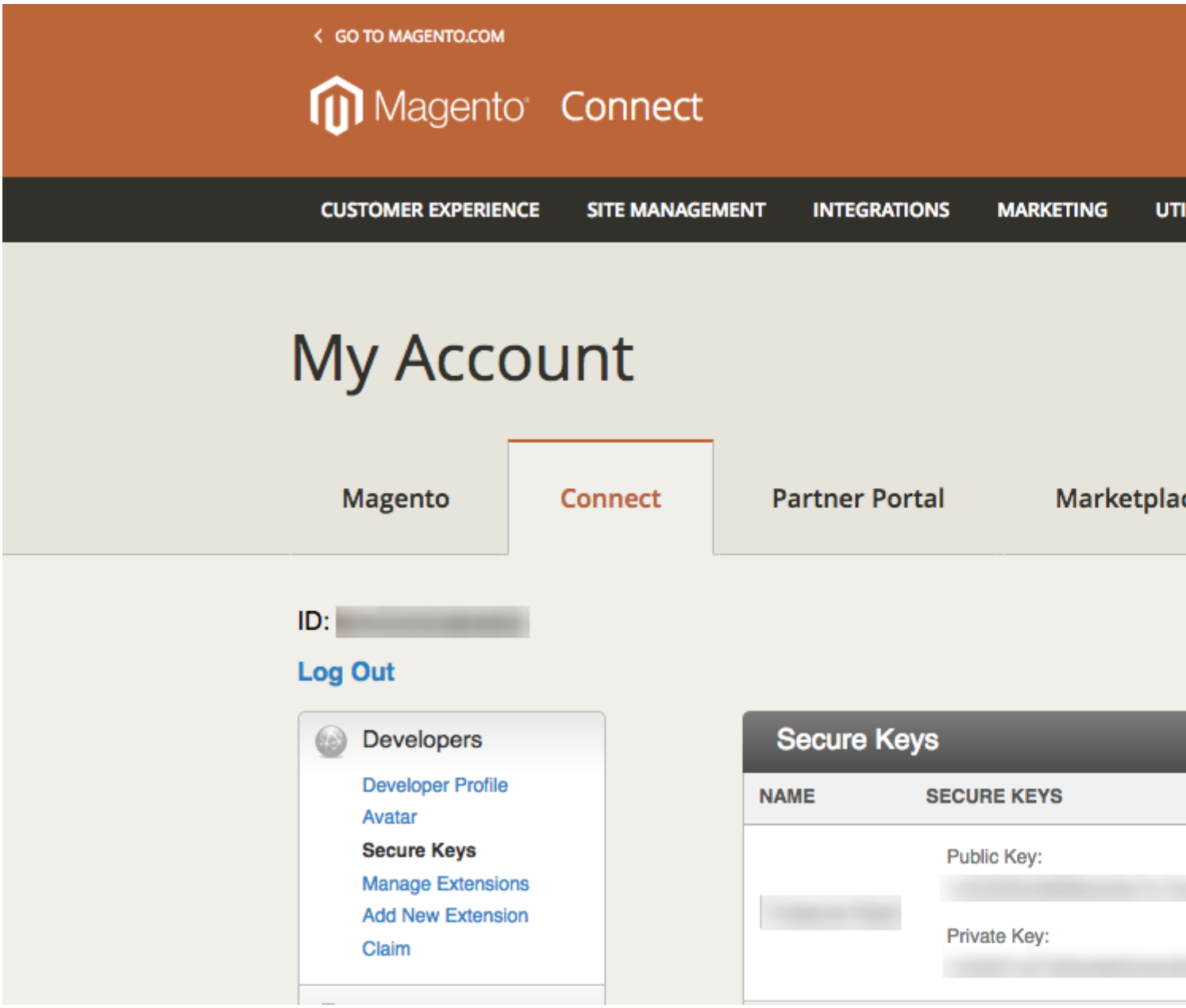
```
cd magento2/
composer install
```

Se il compositore richiede l'autenticazione come di seguito:

```
Loading composer repositories with package information
Installing dependencies (including require-dev) from lock file
- Installing magento/magento-composer-installer (0.1.6)
Downloading: 100%

- Installing braintree/braintree_php (2.39.0)
Downloading: 100%

- Installing justinrainbow/json-schema (1.6.1)
Downloading: 100%

- Installing symfony/console (v2.6.13)
Downloading: 100%

- Installing symfony/process (v2.8.4)
Downloading: 100%

- Installing symfony/finder (v2.8.4)
Downloading: 100%

- Installing seld/jsonlint (1.4.0)
Downloading: 100%

- Installing composer/composer (1.0.0-alpha10)
Downloading: 100%
```

```
- Installing magento/composer (1.0.2)
Authentication required (repo.magento.com):
Username:
Password:
```

Accedi qui https://www.magentocommerce.com/ e usa *la chiave pubblica* come nome ***utente*** e *chiave privata* come ***password*** .



Ora imposta le autorizzazioni su file e directory.

```
sudo chmod -R 755 /var/www/magento2/
sudo chmod -R 777 /var/www/magento2/{pub,var}
```

3. Crea database

Ora accedi al tuo server MySQL con i privilegi di amministratore e crea un database e un utente

per la nuova installazione di Magento2.

```
mysql -u root -p

mysql> CREATE DATABASE magento;
mysql> GRANT ALL ON magento.* TO magento@'localhost' IDENTIFIED BY 'magento';
mysql> FLUSH PRIVILEGES;
mysql> quit
```

4. Configurare Apache VirtualHost e PHP

Crea un file di configurazione Apache per il tuo sito Magento come `/etc/apache2/sites-available/magento2.example.com.conf` e aggiungi il seguente contenuto.

```
<VirtualHost *:80>
   DocumentRoot /var/www/magento2
   ServerName magento2.example.com

   <Directory /var/www/magento2>
       AllowOverride all
   </Directory>
</VirtualHost>
```

Ora abilita virtualhost usando il seguente comando.

```
sudo a2ensite magento2.example.com
```

Assicurati anche di abilitare il modulo di riscrittura di Apache, che è consigliato da Magento.

```
sudo a2enmod rewrite
```

Potresti voler impostare PHP `memory_limit` per evitare la memoria esaurita, cosa che è consigliata anche da Magento.

```
vi /etc/php.ini (find string by press / and type memory_limit)
memory_limit = 768M
```

Dopo aver eseguito tutte le modifiche precedenti, assicurati di riavviare il server Apache.

```
sudo systemctl restart apache2.service
```

5. Installazione dell'applicazione Magento 2

a) Tramite Web Installer

Iniziamo l'installazione di Magento2 usando il programma di installazione web. Accedi alla tua

directory Magento2 sul browser web come di seguito. Ti reindirizzerà alla pagina iniziale dell'installazione.

```
http://magento2.example.com/
```

b) Tramite riga di comando

L'installazione di Magento 2 utilizzando la riga di comando è un miracolo, ha ridotto il tempo di installazione da 10 minuti a 1 minuto. Basta eseguire un comando su una riga.

```
cd /var/www/magento2
php bin/magento setup:install --base-url=http://magento2.example.com/ \
--db-host=localhost --db-name=magento \
--db-user=magento --db-password=magento \
--admin-firstname=Magento --admin-lastname=User --admin-email=user@example.com \
--admin-user=admin --admin-password=admin123 --language=en_US \
--currency=USD --timezone=America/Chicago --cleanup-database --use-rewrites=1
```

6. Pianifica Magento2 Cronjobs

Infine pianifica i backjobund cronjobs per l'installazione di Magento2. Questi cronjobs fanno alcune attività come, reindicizzazione, newsletter, aggiornamento dei tassi di cambio, invio di e-mail automatiche e generazione di sitemap ecc. Per pianificare questi lavori, modifica il file crontab. **www-data** è l'utente di Apache 2, non dovremmo *mai* programmare il cronjob di Magento 2 con il privilegio di root.

```
crontab -u www-data -e
```

Viene visualizzato un editor di testo. (Potrebbe essere necessario scegliere prima un editor di testo.)

```
* * * * * /usr/bin/php /var/www/magento2/bin/magento cron:run | grep -v "Ran jobs by schedule"
>> /var/www/magento2/var/log/magento.cron.log
* * * * * /usr/bin/php /var/www/magento2/update/cron.php >>
/var/www/magento2/var/log/update.cron.log
* * * * * /usr/bin/php /var/www/magento2/bin/magento setup:cron:run >>
/var/www/magento2/var/log/setup.cron.log
```

Leggi Iniziare con Magento2 online: https://riptutorial.com/it/magento2/topic/2279/iniziare-con-magento2

Capitolo 2: Aggiornamento di Magento

Examples

Aggiorna Magento tramite Composer

Controlla la tua attuale versione di Magento

```
php bin/magento --version
```

Ora aggiungi l'ultima versione al tuo compositore.

```
composer require magento/product-community-edition 2.1.6 --no-update
```

Esegui compositore di aggiornamento Questo ti chiederà il nome utente e la password di prendere le tue credenziali dal tuo account di mercato.

```
composer update
```

Questo inizierà il processo per iniziare a scaricare e aggiornare il tuo Magento

Infine aggiorna il contenuto statico e rimuovi la cartella var

```
rm -rf var/di var/generation
php bin/magento cache:flush
php bin/magento setup:upgrade
php bin/magento setup:di:compile
php bin/magento indexer:reindex
```

Ricontrolla la tua versione di Magento.

Leggi Aggiornamento di Magento online:
https://riptutorial.com/it/magento2/topic/9022/aggiornamento-di-magento

Capitolo 3: Evento e osservatore in Magento 2

Examples

Come usare eventi e osservatori personalizzati?

Passaggio 1: creare il file `events.xml` base alle proprie esigenze in `frontend` , `Backend` o entrambi `YKM/Banner/etc/frontend/events.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="../../../../../../lib/internal/Magento/Framework/Event/etc/events.xsd">

    <event name="controller_action_predispatch">
        <observer name="ykm_banner_before" instance="YKM\Banner\Observer\Help" />
    </event>
</config>
```

Passo 2:

Crea un file Observer `YKM/Banner/Observer/Help.php`

```
<?php
/**
 * Copyright © 2015 Magento. All rights reserved.
 * See COPYING.txt for license details.
 */
namespace Estdevs\Banner\Observer;

use Magento\Framework\Event\ObserverInterface;

class Help implements ObserverInterface
{

    public function execute(\Magento\Framework\Event\Observer $observer) {
       echo "this is good.";
    }
}
```

Leggi Evento e osservatore in Magento 2 online:
https://riptutorial.com/it/magento2/topic/5277/evento-e-osservatore-in-magento-2

Capitolo 4: Iniezione di dipendenza

Examples

Sostituzione di argomenti

```
<!-- <moduleDir>/etc/<area>/di.xml -->
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="urn:magento:framework:ObjectManager/etc/config.xsd">
<!-- ... -->
    <type name="Vendor\Namespace\Model\SomeClass">
        <arguments>
            <argument name="object"
xsi:type="object">Vendor\Namespace\Model\SomeOtherClass</argument>
        </arguments>
    </type>
</config>
```

Preferenza di classe

```
<!-- <moduleDir>/etc/<area>/di.xml -->
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="urn:magento:framework:ObjectManager/etc/config.xsd">
<!-- ... -->
    <preference
        for="Vendor\Namespace\Model\Example"
        type="Vendor\Namespace\Model\AnotherExample" />
<!-- ... -->
</config>
```

Sopra l'esempio è una sintassi del modello principale di override.

Ecco una lista di punti che ti descriveranno come renderlo possibile

1. **moduleDir** - Extension directory Come `app/code/custom/extension` here `extension` è la tua directory in cui verranno posizionate tutte le cartelle di estensione necessarie.

2. **area** - area sarà `frontend` o `adminhtml`

    - **frontend** - se l'estensione userà la funzionalità di frontend rispetto a `di.xml` andrà in questa cartella

    - **adminhtml** - se l'estensione utilizzerà la funzionalità di adminpanel rispetto a `di.xml` andrà in questa cartella

    - quindi sarà `app/code/custom/extension/etc/frontend/di.xml` o `app/code/custom/extension/etc/adminhtml/di.xml`

    - Se si desidera utilizzare entrambe le funzionalità, il file `di.xml` verrà diretto nella cartella `etc` non sarà necessario inserirlo nella cartella `frontend` o `adminhtml` . Mi piace - `app/code/custom/extension/etc/di.xml`

3. **for = "Vendor \ Namespace \ Model \ Example"** qui, il percorso del file che sovrascriverà la funzionalità della funzione desiderata.

4. **type = "Vendor \ Namespace \ Model \ AnotherExample"** qui, il percorso del file che fornirà funzioni che sovrascriveranno `step - 3`

Costruttore di iniezione

```
/**
 * @var \Vendor\Module\Helper\Data
 */
protected $customHelper;

/**
 * Constructor call
 * @param \Vendor\Module\Helper\Data $customHelper
 */
public function __construct(
    \Vendor\Module\Helper\Data $customHelper
)
{
    $this->customHelper = $customHelper;
    parent::__construct();
}
```

Leggi Iniezione di dipendenza online: https://riptutorial.com/it/magento2/topic/2998/iniezione-di-dipendenza

Capitolo 5: Magento 2 Comandi per l'uso quotidiano

Osservazioni

Tutti i comandi possono essere eseguiti scrivendo solo una parte di essi.

Per esempio:

- `php bin/magento cache:flush` può essere tradotto in:
  - `php bin/magento c:f`
  - `php bin/magento ca:f`
  - `php bin/magento c:fl`
  - `php bin/magento cache:f`
  - `php bin/magento c:flush`
  - eccetera.

Puoi scrivere qualsiasi parte, e se non è ambiguo, saprà automaticamente quale vuoi.

Examples

compilazione del codice

```
php bin/magento setup:di:compile
```

Potrebbe essere necessario eliminare var / di (inclusa la cartella) per poter eseguire la compilazione.

```
rm -rf var/di
```

Cache a livello

Svuota tutta la cache di Magento

```
php bin/magento cache:clean
php bin/magento cache:flush
```

Controlla lo stato della cache

```
php bin/magento cache:status
```

Abilita estensioni personalizzate o di terze parti

Abilita e aggiorna la configurazione

```
php bin/magento module:enable YKM_Custom
php bin/magento setup:upgrade
```

Disabilita il modulo

```
php bin/magento module:disable YKM_Custom
```

Un altro *script di disinstallazione di* One- *module viene eseguito e l'intero modulo viene eliminato in seguito. Solo i moduli installati tramite Composer possono essere disinstallati.*

```
php bin/magento module:uninstall YKM_Custom
```

Mostra l'elenco dei moduli abilitati e disabilitati

```
php bin/magento module:status
```

Aggiorna lo schema e i dati del database:

```
php bin/magento setup:upgrade
```

Per vedere tutti i comandi disponibili

```
php bin/magento
```

Elenco generale dei comandi per Magento 2

```
php bin/magento setup:upgrade                    => Setup Upgrade
php bin/magento setup:di:compile                 => Setup: Compile
php bin/magento indexer:reindex                  => Reindex
php bin/magento cache:flush                      => Clear Cache
php bin/magento deploy:mode:set developer        => Enable Developer Mode Magento
(developer/production)
php bin/magento deploy:mode:show                 => Show Current Mode Magento
php bin/magento module:status                    => Module: Status
php bin/magento module:disable MODULE_NAME       => Module: Disable
php bin/magento module:enable MODULE_NAME        => Module: Enable
php bin/magento module:uninstall MODULE_NAME     => Module: Uninstall
php bin/magento cron:run                         => Cronjob: Run
```

Leggi Magento 2 Comandi per l'uso quotidiano online:
https://riptutorial.com/it/magento2/topic/5368/magento-2-comandi-per-l-uso-quotidiano

Capitolo 6: Ottieni prodotti dal database

Examples

Ottieni prodotti utilizzando il repository di prodotti

Per ottenere prodotti dal database, è necessario utilizzare il modello di progettazione del repository di Magento 2. Ogni modulo può essere fornito in bundle con i propri repository e il modulo Catalogo prodotti non è diverso.

È possibile utilizzare l' integrazione delle dipendenze nella classe per accedere al repository. Un esempio di lavoro sarebbe simile a questo:

```
class Example
{
    /**
     * @var \Magento\Catalog\Model\ProductRepository
     */
    protected $productRepository;

    /**
     * @param \Magento\Catalog\Model\ProductRepository $productRepository
     */
    public function __construct(
        \Magento\Catalog\Model\ProductRepository $productRepository
    ) {
        $this->productRepository = $productRepository;
    }

    /**
     * Get product by ID
     * @return \Magento\Catalog\Api\Data\ProductInterface
     * @throws \Magento\Framework\Exception\NoSuchEntityException
     */
    public function getProductById(int $productId)
    {
        return $this->productRepository->getById($productId);
    }
}
```

Un repository ha più funzionalità, come il salvataggio o l'eliminazione di un prodotto, oltre a ottenere un elenco di prodotti e l'utilizzo di un filtro, ma questo va oltre lo scopo di questo esempio.

Leggi Ottieni prodotti dal database online: https://riptutorial.com/it/magento2/topic/6459/ottieni-prodotti-dal-database

Capitolo 7: Ottimizzazione di Magento 2

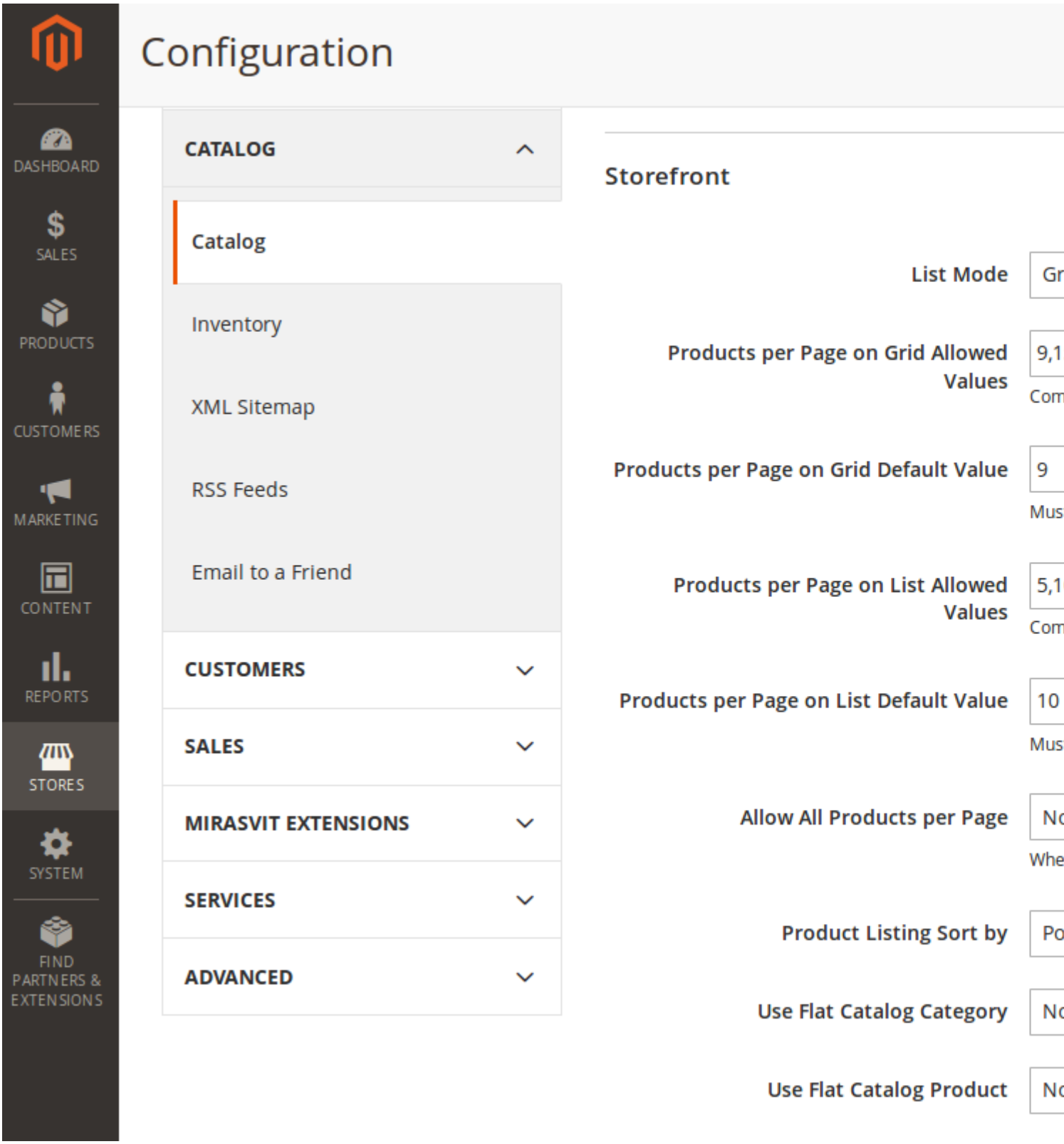
Examples

Configurazioni per ottimizzare

1. Abilitare categorie e prodotti piatti

Uno dei principali motivi dei problemi di velocità di Magento con la velocità di lettura del database. Per fissare la velocità di lettura del database è necessario abilitare **Flat Catalog** . Questo ridurrà il numero di join di database effettuati durante la visualizzazione dei prodotti e, a causa di ciò, la complessità della query MySQL verrà ridotta.

Vai al back-end: **STORES> Configuration> CATALOG> Catalog> Usa Flat Catalog Category** e inserisci **Yes** .

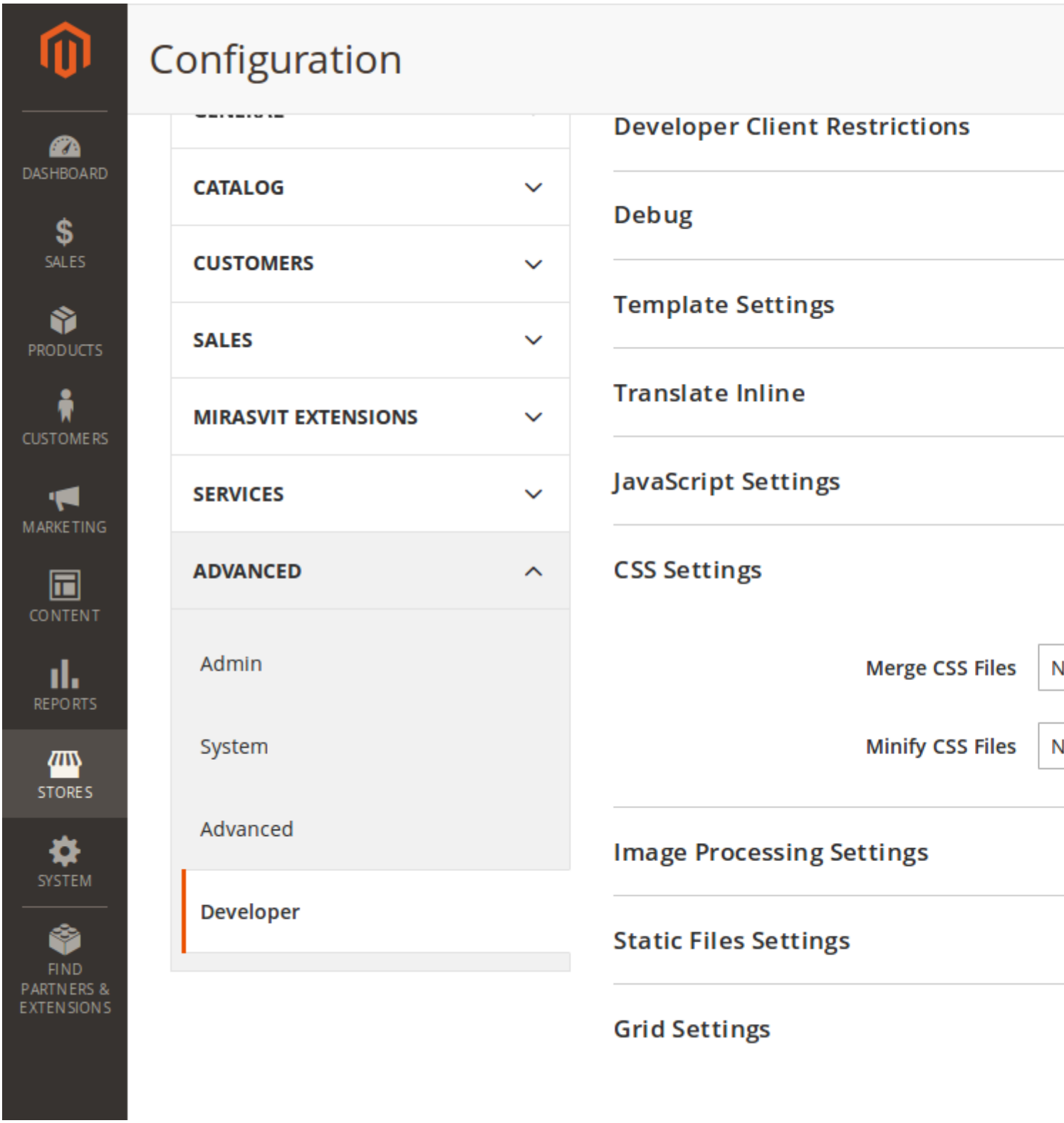


2. Unisci file CSS e JS

Il prossimo passo da seguire è unire e minimizzare i file CSS e Javascript, il che significa rendere la pagina web il più leggera possibile per il caricamento veloce. Si prega di mettere Magento 2 in modalità di **produzione** .

Vai al back-end: **STORES> Configurazione> ADVANCED> Developer> Impostazioni CSS** e

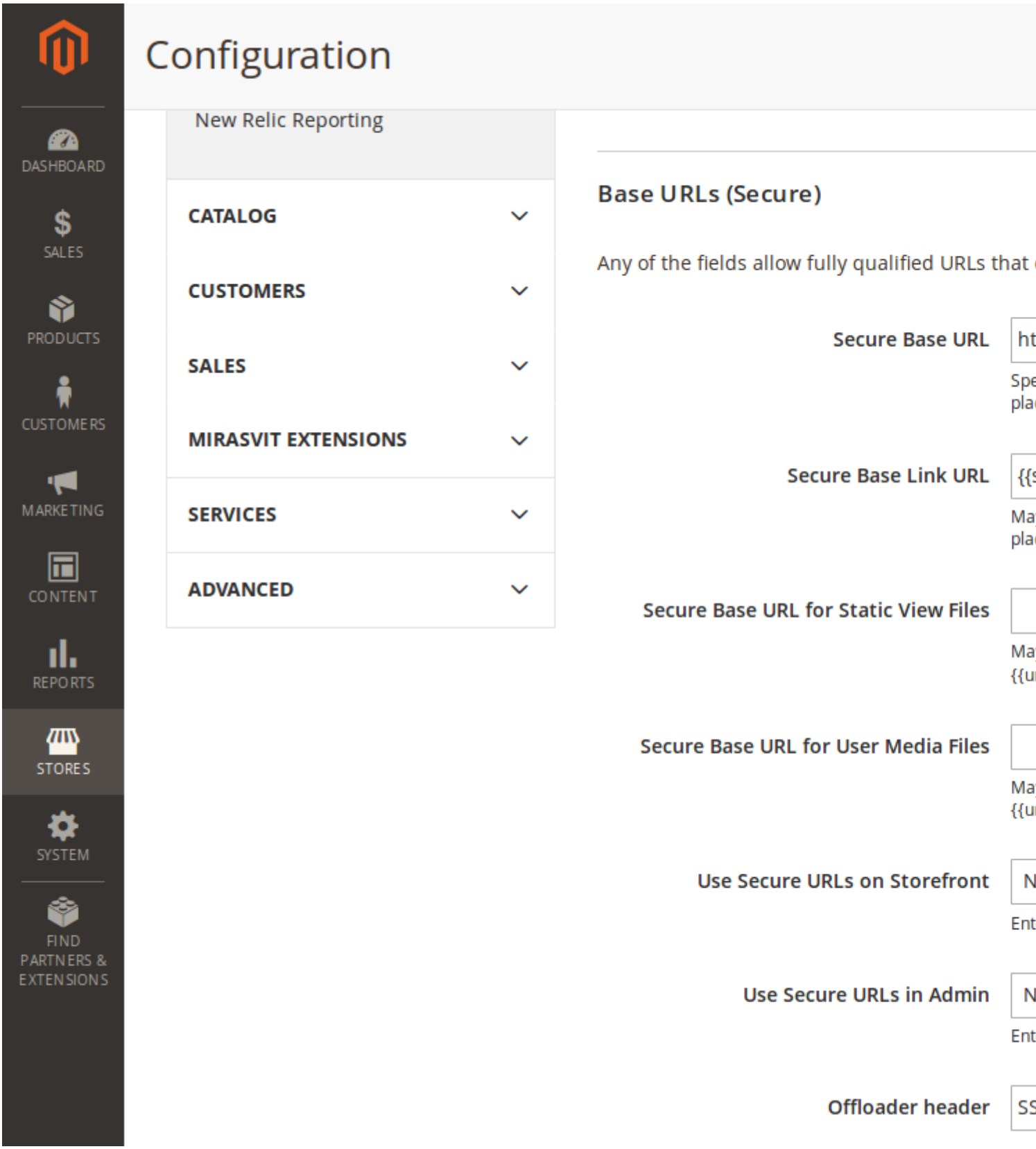
metti i **file Merge CSS** e **Minify CSS** come **yes** .



3. Rete di consegna dei contenuti

CDN, o Content Delivery Network, è un sistema interconnesso di server di cache che utilizzano la vicinanza geografica come criteri per la distribuzione di contenuti Web e di conseguenza aiuta i visitatori a caricare più velocemente le pagine.

Una delle funzionalità di Magento 2 è il supporto immediato di CDN ed è qui che puoi trovare la configurazione per questo: **STORES> GENERAL> Configuration> Web> Base URL (Secure)** e inserisci qui i tuoi URL CDN HTTPS e lascia che i tuoi clienti godano di una rapida velocità di caricamento.



4. Caching

Nel **sistema> Gestione cache** abilita la cache.

Cache Management

Flush Cache Storage

Refresh Submit 13 records found

Cache Type	Description	Tags
Configuration	Various XML configurations that were collected across modules and merged	CONF
Layouts	Layout building instructions	LAYOU
Blocks HTML output	Page blocks HTML	BLOCK
Collections Data	Collection data files	COLLE
Reflection Data	API interfaces reflection data	REFLE
Database DDL operations	Results of DDL queries, such as describing tables or indexes	DB_DI
EAV types and attributes	Entity types declaration cache	EAV
Customer Notification	Customer Notification	CUSTO
Page Cache	Full page caching	FPC
Integrations Configuration	Integration configuration file	INTEG
Integrations API Configuration	Integrations API configuration file	INTEG
Translations	Translation files	TRANS
Web Services Configuration	REST and SOAP configurations, generated WSDL file	WEBS

Additional Cache Management

Flush Catalog Images Cache — Pregenerated product images files

Flush JavaScript/CSS Cache — Themes JavaScript and CSS files combined to one file

Flush Static Files Cache — Preprocessed view files and static files

https://riptutorial.com/it/magento2/topic/10177/ottimizzazione-di-magento-2

Capitolo 8: Prodotti configurabili e loro varianti.

Examples

Ottieni un prodotto per i genitori e i loro figli.

Qui ti mostrerò come recuperare

1. Tutti i genitori (prodotti configuarble)
2. Un prodotto genitore e tutti i suoi figli.

Ottieni un prodotto per i genitori.

Inizieremo creando una classe semplice che ottiene tutti i nostri genitori (prodotti configurabili)

```
<?php

namespace Test\Test\Controller\Test;

use Magento\Framework\App\Action\Context;


class Products extends \Magento\Framework\App\Action\Action
{

        public function __construct(
        \Magento\Catalog\Model\ResourceModel\Product\CollectionFactory $_product_res_fac
        )
        {
            $this->_product_res_fac = $_product_res_fac;
        }

        public function getParentProducts()
        {
            return $this->_product_res_fac->create()->addAttributeToSelect('*')-
>addAttributeToFilter('type_id', ['eq' => 'configurable']);
        }

    }
```

Come si vede sopra la nostra funzione getParentProducts restituirà ora tutti i prodotti configuarabili che attualmente abbiamo nel nostro sistema.

Ottieni prodotti per genitori e figli.

Qui per prima cosa prendiamo il nostro prodotto genitore e otterremo tutti i prodotti per bambini di questo genitore.

```
<?php
```

```
namespace Test\Test\Controller\Test;

use Magento\Framework\App\Action\Context;


class Products extends \Magento\Framework\App\Action\Action
{

        public function __construct(
        \Magento\Catalog\Model\Product $productModel
        )
        {
            $this->product= $productModel;
        }

        public function getParentProduct()
        {
            return $this->product->load("a product entity id goes here")
        }

        public function getChildProducts()
        {
            $_children = $this->getParentProduct()->getTypeInstance()->getUsedProducts($this-
>getParentProduct());
        }

    }
```

La funzione getChildProducts ora restituisce una collezione per bambini in modo da poterla eseguire attraverso un ciclo foreach e ottenere tutti gli attributi del prodotto che potrebbero esserci.

Leggi Prodotti configurabili e loro varianti. online: https://riptutorial.com/it/magento2/topic/10790/prodotti-configurabili-e-loro-varianti-

Capitolo 9: Sostituisci il language pack i18n

Sintassi

- **<Spazio dei nomi dei fornitori>** - Qui spazio dei nomi del tema personalizzato del fornitore o dello spazio dei nomi dei temi integrato **IE** `Magento/Luma` Here `luma` è `vendor namespace`
- **<directory del pacchetto lingua>** - Qui la directory del pacchetto di lingua come `en_us` o `nl_nl` o `en_gb`
- **<descrizione del pacchetto linguistico>** - Qui aggiungi la descrizione del pacchetto come `English Us Package`
- **<codice pacchetto lingua>** - Qui il codice del pacchetto lingua **IE** `en_US` o `nl_NL` o `en_GB`

Osservazioni

Dopo aver creato sopra i file e le directory `language_package_code.csv` andrà nella directory `Vendor Namespace`

Esempio

`/app/i18n/luma/en_us/en_US.csv`

o

`/app/i18n/luma/en_gb/en_GB.csv`

o

`/app/i18n/luma/nl_NL/nl_NL.csv`

Examples

Esempio di sintassi del pacchetto di lingua i18n di override

/ app / i18n / <Spazio dei nomi del fornitore> / <directory del pacchetto lingua> /composer.json

```
{
    "name": "<vendor namespance>/<language package directory>",
    "description": "<language package description>",
    "version": "100.0.1",
    "license": [
        "OSL-3.0",
        "AFL-3.0"
    ],
    "require": {
        "magento/framework": "100.0.*"
    },
```

```
    "type": "magento2-language",
    "autoload": {
        "files": [
            "registration.php"
        ]
    }
}
```

/ app / i18n / <Spazio dei nomi del fornitore> / <Language Pack> /language.xml

```
<?xml version="1.0"?>
<language xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="urn:magento:framework:App/Language/package.xsd">
    <code><language package code></code>
    <vendor><vendor namespace></vendor>
    <package><language package directory></package>
</language>
```

/ app / i18n / <Spazio dei nomi del fornitore> / <pacchetto lingua> /registration.php

```
<?php
\Magento\Framework\Component\ComponentRegistrar::register(
    \Magento\Framework\Component\ComponentRegistrar::LANGUAGE,
    '<vendor namespance>_<language package directory>',
    __DIR__
);
```

Leggi Sostituisci il language pack i18n online:
https://riptutorial.com/it/magento2/topic/10789/sostituisci-il-language-pack-i18n

Capitolo 10: Struttura del modulo

Examples

Struttura del modulo del catalogo

Per ora penso che il modulo catalogo contenga quasi tutto ciò che puoi aggiungere a un modulo.

- **Api** : contiene i contratti di servizio. Un insieme di interfacce che non dovrebbero essere cambiate a meno che la versione secondaria non cambi. Non obbligatorio per un modulo personalizzato ma piacevole da avere per le estensioni commerciali.
    - **Dati** - Interfacce dati. Ogni interfaccia deve avere un modello che la implementa (esempio: interfaccia per modello di prodotto)
    - **ProductRepositoryInterface.php** - interfacce per repository (deve anche avere un'implementazione)
    - ... - altri come sopra
- **Blocchi** - blocchi utilizzati nel layout per frontend e backend
    - **Adminhtml** - blocchi utilizzati per il back-end
    - **Categoria** - blocchi relativi al frontend. Può essere annidato in tutte le cartelle che vuoi, ma non è obbligatorio
    - ... - come sopra
- **Console** - cartella contenente i comandi cli
- **Controller** : contiene **controller** frontend e backend
    - **Adminhtml** - controller back-end
    - **Categoria** : controllori correlati frontend. Può essere annidato in tutte le cartelle che vuoi, ma non è obbligatorio
    - ... - come sopra.
- **Cron** - codice che dovrebbe essere eseguito tramite cron
- **etc** - contiene i file xml di configurazione del modulo
    - **frontend** - contiene i file di configurazione caricati solo sul frontend
    - **adminhtml** - contiene i file di configurazione caricati solo sul back-end
    - **webapi_rest** - contiene i file di configurazione caricati solo per il resto API
    - **webapi_soapt** : contiene i file di configurazione caricati solo per l'API SOAP
    - **acl.xml** - Definizioni ACL
    - **catalog_attributes.xml** : attributi predefiniti per le entità del catalogo.
    - **catalog_attributes.xsd** - schema di convalida per il file sopra.
    - **config.xml** : valori predefiniti per le impostazioni di configurazione
    - **crontab.xml** - cron job scheduling
    - **di.xml** - preferenze di iniezione delle dipendenze. (può anche risiedere in adminhtml, frontend, webapi_ *)
    - **events.xml** - dichiarazione degli osservatori per gli eventi (può anche risiedere in adminhtml, frontend)
    - **indexer.xml** - impostazioni per diversi indici che devono essere eseguiti quando i dati cambiano
    - **module.xml** - il file di dichiarazione del modulo

    - **product_ *** - impostazioni relative al prodotto.
    - **webapi.xml** - percorsi di dichiarazione webapi.
    - **widget.xml** - dichiarazioni di widget.
- **Helper** : diversi helper del modulo
- **i18n** - file di traduzione in lingua
- **Modello** - modelli, semplice come quello. possono essere annidati in tutte le cartelle che vuoi, ma non è obbligatorio.
- **Observer** - classi di osservatori di eventi
- **Plugin** - `around|before|after` plugin per diversi metodi pubblici.
- **Prezzi** - classi relative ai prezzi. Questo è specifico del modulo. Puoi avere tutte le cartelle che vuoi in questo modo se non vuoi metterle nella cartella dei modelli.
- **Installazione** - installa / aggiorna i file correlati (installando schema e dati di aggiornamento)
- **Test** : test unitari
- Classi relative ai componenti **Ui** - ui.
- **view** - la parte relativa a html. La **V** in MVC.
    - **adminhtml** - file relativi **all'amministratore**
        - **layout** - layout xml per adminhtml
        - **templates** - template phtml per adminhtml
        - **ui_compoenent** - file relativi ai componenti ui (dichiarazione)
        - **web** - assets (js, images)
        - **requirejs-config.js** - configuration for require.js
    - **base** - file usati sia per frontend che per backend.
        - può avere la stessa struttura di sottocartella di adminhtml
    - **frontend** - file frontend correlati
        - può avere la stessa struttura di sottocartella di adminhtml
- **compositore.json** - non obbligatorio, ma piacevole da avere se si distribuisce il modulo
- **registration.php** : il file di registrazione del modulo.
- **Licenza * .txt, readme.md** - sai cosa significa. Non sono obbligatori

Leggi Struttura del modulo online: https://riptutorial.com/it/magento2/topic/4838/struttura-del-modulo

Capitolo 11: Tema personalizzato

Osservazioni

tema `luma` come genitore

```
{
    "name": "magento/luma",
    "description": "N/A",
    "require": {
        "php": "~5.5.0|~5.6.0|~7.0.0",
        "magento/theme-luma": "100.0.*",
        "magento/framework": "100.0.*"
    },
    "type": "magento2-theme",
    "version": "100.0.1",
    "license": [
        "OSL-3.0",
        "AFL-3.0"
    ],
    "autoload": {
        "files": [
            "registration.php"
        ]
    }
}
```

alla fine

Esegui `php bin/magento setup:upgrade` questo comando dopo che i comandi di seguito sono necessari anche a volte

- `php bin/magento setup:static-content:deploy <language_pack_1> <language_pack_2> ... <language_pack_n>`
    - **<language_pack>** : `en_US nl_NL en_GB` ecc
- `php bin/magento cache:flush` o `php bin/magento cache:clean`

Examples

Tema di esempio

Theme.xml

`app/design/frontend/Magento/mytheme/theme.xml`

```
<theme xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="urn:magento:framework:Config/etc/theme.xsd">
     <title>My theme</title> <!-- your theme's name -->
     <parent>Magento/blank</parent> <!-- the parent theme, in case your theme inherits from an
existing theme -->
```

```
     <media>
         <preview_image>media/preview.jpg</preview_image> <!-- the path to your theme's
preview image -->
     </media>
 </theme>
```

app/design/frontend/Magento/mytheme/composer.json

```
{
    "name": "magento/theme-frontend-blank",
    "description": "N/A",
    "require": {
        "php": "~5.5.0|~5.6.0|~7.0.0",
        "magento/theme-frontend-blank": "100.0.*",
        "magento/framework": "100.0.*"
    },
    "type": "magento2-theme",
    "version": "100.0.1",
    "license": [
        "OSL-3.0",
        "AFL-3.0"
    ],
    "autoload": {
        "files": [
            "registration.php"
        ]
    }
}
```

app/design/frontend/Magento/mytheme/registration.php

```
<?php
/**
* Copyright © 2015 Magento. All rights reserved.
* See COPYING.txt for license details.
*/
\Magento\Framework\Component\ComponentRegistrar::register(
    \Magento\Framework\Component\ComponentRegistrar::THEME,
    'frontend/Magento/mytheme',
    __DIR__
);
```

alla fine

```
php bin/magento setup:upgrade
```

Leggi Tema personalizzato online: https://riptutorial.com/it/magento2/topic/6244/tema-personalizzato

Capitolo 12: Utilizzo dell'iniezione delle dipendenze per riscrivere l'oggetto

Osservazioni

https://gielberkers.com/magento-2-why-use-rewrites-when-you-can-use-plugins/

http://devdocs.magento.com/guides/v2.0/extension-dev-guide/plugins.html

Examples

Alcuni modi per modificare una funzione in Magento 2

Riscrivi la classe

File: `Namespace/ModuleName/etc/di.xml`

```
<?xml version="1.0"?>
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="urn:magento:framework:ObjectManager/etc/config.xsd">
   <preference for="Magento\Catalog\Controller\Product\View"
type="Namespace\ModuleName\Controller\Product\View" />
</config>
```

File: `Namespace\ModuleName\Controller\Product\View.php`

```
class View extends \Magento\Catalog\Block\Product\View
{
    ///Code logic here
}
```

Plugin in oggetto.

File: `Namespace/ModuleName/etc/di.xml`

```
    <?xml version="1.0"?>
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="urn:magento:framework:ObjectManager/etc/config.xsd">
        <type name="Magento\Catalog\Model\Product">
            <plugin name="name_of_plugin"
type="Namespace\ModuleName\Plugin\Catalog\Model\Product" sortOrder="1" disabled="false" />
        </type>
    </config>
```

File: `Namespace\ModuleName\Plugin\Catalog\Model\Product.php`

```
namespace Namespace\ModuleName\Plugin\Catalog\Model;

class Product
{
    public function beforeSetName(
        \Magento\Catalog\Model\Product $product, string $name)
    {
        /// Code logic here
        return $name;
    }

    public function afterGetName(
        \Magento\Catalog\Model\Product $product, string $name)
    {
         /// Code logic here
        return $name;
    }

    public function aroundSave(
        \Magento\Catalog\Model\Product $product, \Closure $proceed)
    {
        $this->doSomethingBeforeSave();
        $result = $proceed();
        if ($result) {
            $this->doSomethingAfterSave();
        }
        return $result;
    }
}
```

Leggi Utilizzo dell'iniezione delle dipendenze per riscrivere l'oggetto online: https://riptutorial.com/it/magento2/topic/6283/utilizzo-dell-iniezione-delle-dipendenze-per-riscrivere-l-oggetto

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare con Magento2	4444, Community, ehzawad, Marek Skiba, Niroshan Ranapathi, Priyank, Rafael Corrêa Gomes, Toan Nguyen
2	Aggiornamento di Magento	Priyank
3	Evento e osservatore in Magento 2	Reena Parekh, Yogendra - eCommerce Developer
4	Iniezione di dipendenza	bpoiss, Giel Berkers, Nirav Joshi
5	Magento 2 Comandi per l'uso quotidiano	AlexL, Andrew Stepanchuk, Ankit Shah, belfort1, Jignesh Khunt, matiaslauriti, Yogendra - eCommerce Developer
6	Ottieni prodotti dal database	Giel Berkers, matiaslauriti
7	Ottimizzazione di Magento 2	Rafael Corrêa Gomes
8	Prodotti configurabili e loro varianti.	Anoxy
9	Sostituisci il language pack i18n	Nirav Joshi
10	Struttura del modulo	Akif, belfort1, Giel Berkers, Marius, Tom
11	Tema personalizzato	Nirav Joshi, Qaisar Satti
12	Utilizzo dell'iniezione delle dipendenze per riscrivere l'oggetto	Dmitri Sologoubenko, HoangHieu, Rafael Corrêa Gomes