Magento Development

Scopi del corso:

- **Where is it**: una conoscenza approfondita dell'architettura della piattaforma per uno sviluppo efficiente;
- **Sviluppo backend su Magento**: dalle basi al rilascio di un modulo;
- **Best practices** per l'implementazione di nuove funzionalità e la personalizzazione di quelle esistenti;
- **La dependency injection**, i plugins e gli observers;
- **Integrazione di moduli esterni**, come valutarli e sceglierli;

Competenze richieste:

- html, css, js, php, apache, nginx, mysql, xml

Competenze consigliate:

- *PSR, Oop, Docker, redis, varnish, esperienza anche minima di sviluppo su Magento2*

Presentazioni

Salvatore Capritta

- Co-Founder e CTO @ Synthetic;
- Sviluppatore Certificato Magento;
- Vivo a Milazzo (Sicilia) dove sviluppo e progetto dal 2000.

Di cosa mi occupo:

- **Progetto** le nuove soluzioni e i nuovi prodotti;
- **Analizzo** e introduco le nuove tecnologie per il team;
- **Formo** sui nuovi strumenti i miei compagni di viaggio;
- **Sviluppo** in prima persona alcune parti dei nuovi progetti.

Presentazioni e Aspettative / Richieste

- Valuteremo le competenze in ingresso e il livello medio di conoscenza della piattaforma;
- Ci occuperemo di approfondire quanto possibile e con esempi pratici le potenzialità backend di Magento/Adobe Commerce
- Conoscere le logiche della piattaforma ci permetterà di poterla personalizzare per i nostri casi d’uso.

Scaletta del corso

Giorno 1

Prima Installazione e check delle competenze

Concetti base ed esercitazioni

Giorno 2

Configurazioni

Gestione Contenuti e Catalogo

Creazione Modulo

Creazione Tema

Giorno 3

ORM in Magento2

Dependency Injection

Attributi Prodotto e Data Patch

Giorno 4

Controller e Routing

Plugins e Observers

Cronjobs

Giorno 5

Api, Rest e Graphql

Headless Magento

Giorno 6

Tecniche di sviluppo ed esercitazioni

Best Practices e Troubleshooting

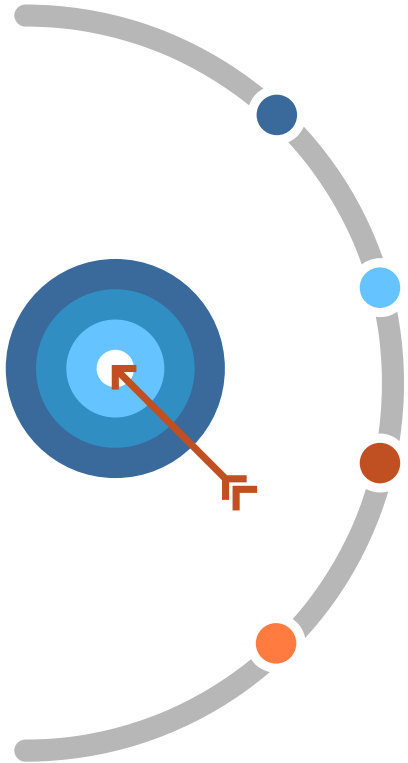
Giorno 1

Prima Installazione e check delle competenze

Concetti base ed esercitazioni

Concetti e requisiti preliminari

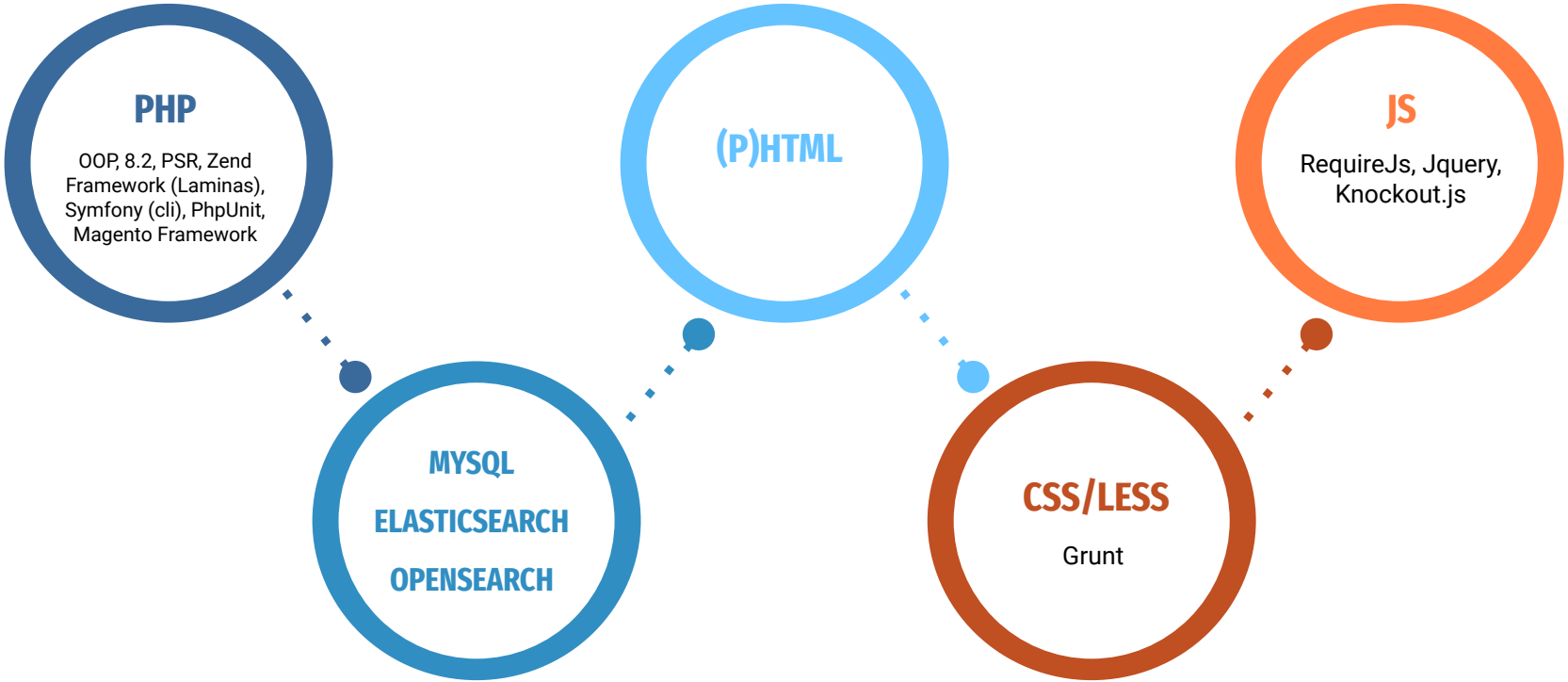
1. Come è installata / si installa la piattaforma;
2. Concetto di componenti e loro sotto-tipi:
    - modulo
    - tema
    - language pack;
3. concetto di tema parent e child;
4. anatomia base del modulo;
5. consapevolezza su architettura e filesystem;
6. configurazioni di base: dove, quali sono e come si gestiscono;
7. conoscere la/le cache.



Introduzione alla Piattaforma

- Magento, è una piattaforma ecommerce di proprietà di Adobe.
- Esiste una versione Open Source (Magento) e una Versione Commerce (Adobe Commerce);
- Esiste anche il Pwa Studio, considerato il nuovo frontend sottoforma di SPA e PWA https://devdocs.magento.com/guides/v2.4/pwa/
- Ha una lunga storia, versione 1 nata dalla Varien -> Ebay -> Adobe: https://it.wikipedia.org/wiki/Magento
- La 2 è uscita a cavallo tra il 2015 e il 2016 e attualmente la versione è arrivata alla 2.4.6 -> https://experienceleague.adobe.com/docs/commerce-operations/release/notes/magento-open-source/2-4-5.html?lang=it
- Documentazione e risorse utili di base:
    - Devdocs:https://developer.adobe.com/commerce/docs/
    - UserGuide: https://experienceleague.adobe.com/docs/commerce-admin/user-guides/home.html?lang=it

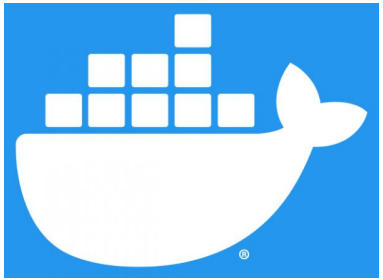
Introduzione alla Piattaforma



Introduzione alla Piattaforma

- Stack per lo sviluppo:
  - Apache / Nginx
  - Php 8.x
  - Mysql
  - Elasticsearch / OpenSearch
  - Redis
  - Varnish (reverse proxy)

Ambienti di Sviluppo



Ambienti di sviluppo:

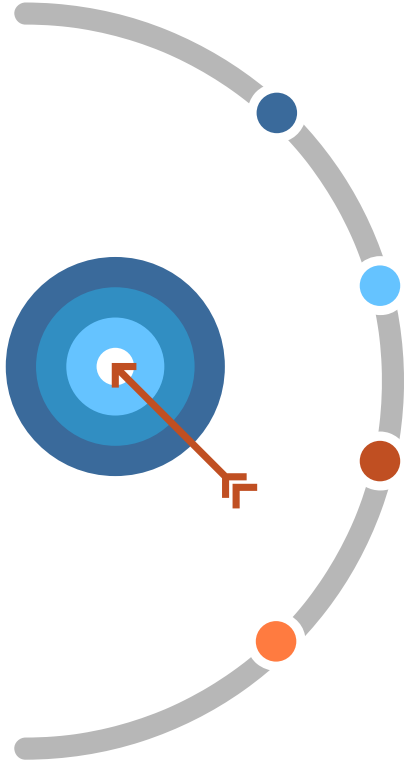
1. **Docker:**
   - **Installing Warden — Warden documentation**
   - Warden per Apple Silicon M1 e successivi
   - markshust/docker-magento: Mark Shust's Docker Configuration for Magento
2. Installazione personalizzata su ambiente Linux
3. https://github.com/weprovide/valet-plus (mac)
4. https://cpriego.github.io/valet-linux/ (linux)

Sistemi operativi supportati:

- mac
- linux
- windows con wsl -> Windows Wsl

Installazione della piattaforma;

- Riferimento ufficiale: On-premises installation overview | Adobe Commerce
- Creazione coppia di chiavi di accesso su marketplace.magento.com
    - dopo il login (o registrazione)
        - ritornare su marketplace.magento.com
        - in alto a destra (nome cognome utente) cliccare su freccetta -> “My Profile”
        - My Products -> Access Keys
        - Create new Access Keys
    - Chiavi demo per il corso:
        - public key (username): 427487c2a1d0f05f7665a22c56cd0e26
        - private key (password): a701077dc99eb0c7b001eb4858572ddc
- Installazione con **composer**;
- **Cli** per comandi di installazione e personalizzazione;
- Warden: Initializing An Environment — Warden documentation
- Cli, bin/magento: bin/magento (Open Source) | Magento 2 Developer Documentation
- Disable 2FA: https://github.com/markshust/magento2-module-disabletwofactorauth
- Sampledata: Install using Composer | Magento 2 Developer Documentation

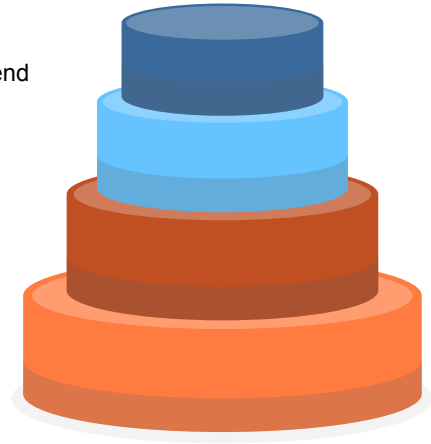


Componenti e sotto-tipi:

- **Tipologie di Componenti**
  - Ogni componente di Magento ha delle caratteristiche obbligatorie e delle altre opzionali.
  - Ogni componente può essere di una di queste tipologie (composer package type):
    - **magento2-module:**
      - sono i moduli che aggiungono funzionalità su Magento;
    - **magento2-theme:**
      - risiedono generalmente su app/design/[area] (frontend o adminhtml)
      - o su vendor/vendorname/theme-[area]-name (Es. vendor/magento/theme-frontend-blank)
      - contengono elementi che definiscono l'aspetto della piattaforma in una delle sue aree.
    - **magento2-language:**
      - costituiscono dei pacchetti contenenti traduzioni delle stringhe presenti nei vari moduli;
      - generalmente vengono creati ed utilizzati per creare versioni localizzate del progetto in varie lingue.
    - magento2-component:
      - componenti di base della piattaforma;
      - un esempio è il "magento2-base" presente su vendor/magento
    - magento2-library:
      - le librerie di base su cui si appoggia la piattaforma;
      - un esempio è presente su vendor/magento/framework

Filesystem e funzionalità

- **app**
  - app/code: qui risiedono i moduli che andremo a sviluppare o ad integrare
  - app/design: frontend o adminhtml: qui risiederanno i temi che definiranno l'aspetto del frontend o del backend
  - app/i18n: cartella destinata a contenere i language packs per le traduzioni;
  - app/etc: cartella contenente le configurazioni principali
- **bin**
  - bin/magento! :)
- **dev**
  - cartella con gli strumenti di sviluppo (come ad esempio grunt per compilare il codice less);
- **generated**
  - in questa cartella verranno creati da magento dei file che saranno il frutto dell'integrazione di plugin, factory e proxies così come richiesti dal codice principale;
  - è un cartella che si può considerare come "temporanea" (tra virgolette)
- **lib**
  - contiene le librerie su cui si basa la piattaforma;
- **pub**
  - la cartella da considerare come la root consigliata sul web server.
  - **pub/media:** cartella contenente immagini e media correlati ai prodotti e generalmente inseriti da backend;
  - **pub/static:** cartella con le risorse statiche generate da magento in fase di compilazione (production mode) o su richiesta (developer mode)
- **setup**
  - cartella con file per installazione da browser (deprecato)
- **var**
  - file temporanei (come log, cache o similari)
- **vendor**
  - la cartella dove composer installerà i pacchetti richiesti tramite composer.json e composer.lock



Giorno 2

Configurazioni

Gestione Contenuti e Catalogo

Creazione Modulo

Creazione Tema

Configurazione

- User Guide: https://docs.magento.com/user-guide/
- Dev Docs: https://devdocs.magento.com/
- dov'è il mio admin? -> bin/magento info:adminuri

Configurazioni ottimali di partenza:

- **Indici e Cron**: https://devdocs.magento.com/guides/v2.4/config-guide/cli/config-cli-subcommands-cron.ht
  - admin -> sistema -> gestione indice -> seleziona tutto -> aggiorna da pianificazione
  - oppure da cli: Manage the indexers | Magento 2 Developer Documentation
  - bin/magento cron:install
- **xsd**: bin/magento dev:urn-catalog:generate .idea/misc.xml
  - URN highlighter | Magento 2 Developer Documentation
  - (su warden/docker, generiamo nella root e poi copiamo dall'host dentro la cartella .idea)
- **Lingua italiana**
  - repo qui: https://github.com/mageplaza/magento-2-italian-language-pack
  - composer require mageplaza/magento-2-italian-language-pack:dev-master
  - bin/magento cache:clean
  - admin -> stores -> settings -> configuration -> general -> locale options
  - admin -> account settings
  - i18n -> personalizzazione della lingua in frontend -> tema

Configurazioni di partenza

- **Indici e Cron**: https://devdocs.magento.com/guides/v2.4/config-guide/cli/config-cli-subcommands-cron.ht
  - admin -> sistema -> gestione indice o da cli: Manage the indexers | Magento 2 Developer Documentation
- **Lingua italiana**
  - repo qui: https://github.com/mageplaza/magento-2-italian-language-pack
  - admin -> stores -> settings -> configuration -> general -> locale options (e admin -> account settings)
- **Generali**
  - Configurare dati generali da Store->Configuration->General (ragione sociale, nazione, titolare)
- **Pagamento**
  - vendite -> impostazioni metodi di pagamento
- **Spedizione**
  - vendite -> impostazioni metodi di spedizione
  - store -> configuration -> sales -> multishipping settings -> allow shipping to multiple address > no
- **Stock**
  - Stores->Configuration->Catalog->Inventory->Product Stock Options-> Minimum Qty Allowed in Shopping Cart ->Add Customer Group -> impostare 1
- **Sicurezza**
  - Stores->Configuration->Advanced->Admin->Security->Admin Session Lifetime (seconds)->Impostare almeno a 7200 (di default è 900) e durata password.
  - Stores->Configuration->Advanced->Admin->Security->Modifica della password (consigliata)
- **SMTP**
  - https://docs.mageplaza.com/smtp-m2/

Configurazioni

Backend

- **Generali**
  - Configurare dati generali da Store->Configuration->General (ragione sociale, nazione, titolare)
- **Pagamento**
  - vendite -> impostazioni metodi di pagamento
  - metodi sandbox
- **Spedizione**
  - vendite -> impostazioni metodi di spedizione
  - table rates generator -> Table Rates Generator
  - store -> configuration -> sales -> multishipping settings -> allow shipping to multiple address > no
- **Admin user**
  - system -> user role
  - system -> add user associando il nuovo user role
    - bin/magento admin:user:create
- **Email transazionali:**
  - inserire templates per le email di conferma ordine e pagamento fallito e impostare indirizzi email del sito
  - marketing -> email templates
  - store configuration -> sales email

Configurazioni

- **Stock**
  - Stores->Configuration->Catalog->Inventory->Product Stock Options-> Minimum Qty Allowed in Shopping Cart ->Add Customer Group -> impostare 1
- **Seo**
  - Stores->Configuration->Catalog->Catalog->Search Engine Optimization->Suffisso url categoria -> rimuovere
  - Stores->Configuration->Catalog->Catalog->Search Engine Optimization->Suffisso url prodotto -> rimuovere
  - Stores->Configuration->Catalog->Catalog->Search Engine Optimization->Use Canonical Link Meta Tag For Categories
  - Stores->Configuration->Catalog->Catalog->Search Engine Optimization->Use Canonical Link Meta Tag For Products
- **Sicurezza**
  - Stores->Configuration->Advanced->Admin->Security->Admin Session Lifetime (seconds)->Impostare almeno a 7200 (di default è 900) e durata password.
  - Stores->Configuration->Advanced->Admin->Security->Modifica della password (consigliata)
  - Negozi -> Configurazioni -> Clienti-> nome e opzioni indirizzo -> mostra prefisso -> opzionale
  - Negozi -> Configurazioni -> Clienti-> carrello persistente -> abilita la persistenza -> sì
  - Negozi -> Configurazioni -> Clienti -> configurazione cliente -> opzioni password -> numero di classi di carattere richieste -> 2
  - Negozi->Configurazioni->Clienti->Configurazione Cliente->Opzioni Password-> Tempo minimo tra le richieste di Ripristino Password-> 0
  - Per possibili problemi Seo togliere il referer dal url del login
    - Negozi->Configurazioni->Clienti->Configurazione Cliente->Opzioni login->Reindirizzare il cliente alla Dashboard dell'Account dopo il login-> Impostare su Sì

Configurazioni

- **Tasse e IVA**
  - Negozi -> Tasse -> Zone e aliquote:
    - cancellare quelle di esempio (usa) e inserire "IVA" con percentuale al 22%, zipcode=*, stato italia
  - Negozi -> Tasse -> Regole Fiscali
    - inserire regola con stesso nome IVA IT e associazione ad aliquota
  - Negozi -> configurazione -> Vendite -> Tasse
    - uniformare visualizzazione prezzi in carrello e in ordini e email
  - Impostare metodo di spedizione default:
    - Negozi->Configurazioni->Vendite->Tassa -> Calcolo tassa in base alla destinazione predefinito -> paese di default -> italia
  - per approfondimenti: Tax - User Guide - Magento
- **Sitemap**
  - Negozi -> Configurazioni -> Catalogo -> Xml Sitemap -> Impostazioni di generazione -> Abilitato -> Sì
  - Negozi -> Configurazioni -> Catalogo -> Xml Sitemap -> Impostazioni invio del motore di ricerca -> Attiva Invio al Robots.txt -> sì
  - Marketing -> Ricerca & Seo -> Sitemap -> Aggiungi sitemap -> con nome file "sitemap.xml" e percorso "/" -> salva e genera
  - Aggiungerla in Search Console -> https://search.google.com/search-console
- **SMTP**
  - https://docs.mageplaza.com/smtp-m2/

Configurazioni

Ma dove stiamo salvando questi dati?

- warden db connect
    - (con e senza system value)
    - valori di default (etc/config.xml)
    - valori personalizzati (in db)
    - impostazione valori su etc/adminhtml/system.xml
- testiamo il nostro store
    - registriamoci
    - effettuiamo un ordine
    - cambiamo il contenuto dell’email di vendita
- un ultimo sguardo all’admin:
    - catalog
    - contenuti
        - pagine
        - blocchi
        - widgets

Contenuti e Catalogo

Cosa possiamo gestire direttamente da admin?

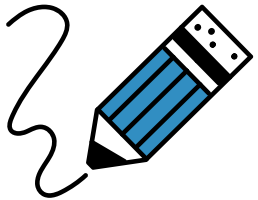
Gestione Contenuti - User Guide - Magento

- **contenuti**
  - pagine
  - blocchi
    - posizioni: Positioning Blocks
  - widgets
  - page layout e layout updates
- **catalogo**
  - categorie
  - prodotti

Concetto di tema parent e child

Creiamo un tema,

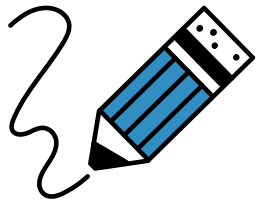
Devdocs: **Create a new storefront theme | Magento 2 Developer Documentation**



- creazione cartella
    - app/design/frontend/Accenture/Luma
    - impostazione file obbligatori (theme.xml, registration.php, composer.json)
    - opzionale -> etc/view.xml
    - web/images/logo.svg -> automaticamente riconosciuto come logo del tema
- impostazioni da admin, applichiamo il tema appena creato: Applicazione del Tema
- impostiamo un sito multilingua: Multiple websites or stores | Magento 2 Developer Documentation
    - Set up multiple websites, stores, and store views in the Admin | Magento 2 Developer Documentation
        - website -> azienda (politiche generali, di prezzo, etc...)
        - store -> negozio (con differenze di catalogo, es. negozio uomo, negozio donna, etc...)
        - storeview -> commessi (chi parla inglese, chi francese, etc...)
    - aggiungiamo una store view (es. per la versione inglese da Stores -> All Stores)
    - attiviamo il codice store view nell'url da: Stores > Settings > Configuration > General > Web -> Add Store Code to Urls.
    - associamo la corretta lingua alla store view da -> admin -> stores -> settings -> configuration -> general -> locale options

Concetto di tema parent e child

Processo di elaborazione dei due principali “elementi frontend” da parte di Magento2 a seguito di una richiesta:

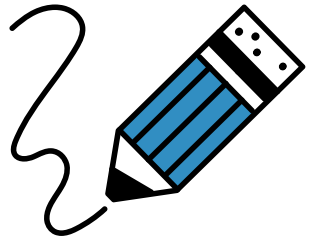


- **Recupero template .phtml:**
  - cerca nel tema applicato;
  - se non trova, cerca in eventuale tema parent, fino a completamento della “catena” dei temi;
  - (se non trova) cerca nel modulo di magento che gestisce la funzionalità / sezione

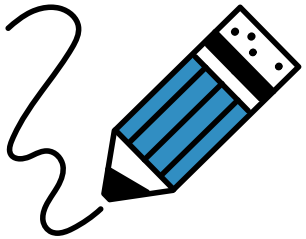
- **Elaborazione layout:**
  - per i layout viene effettuato un merge seguendo questo schema:
    - si parte dal layout originario (es. catalog_category_view.xml nel modulo Magento_Catalog)
    - seguendo l’ordine di esecuzione dei moduli (config.php) vengono cercati altri layout xml con stesso nome
    - seguendo l’ordine dichiarato con la relazione parent - child dei temi vengono cercati altri layout xml con stesso nome in tutti i temi coinvolti (partendo dal primo ancestor fino al tema applicato)
    - il layout finale, frutto di tutti i merge e dell’esecuzione di tutti gli eventuali comandi inseriti, viene utilizzato.
- **ViewModel:**
  - un “nuovo” sistema per gestire business logic necessaria alle view (e non solo) in maniera più snella e flessibile rispetto al classico binomio “block -> template”

Personalizzazione Frontend

- **Magento Modes:** About Magento modes | Magento 2 Developer Documentation
- **I mattoncini della personalizzazione frontend:**
  - layout, templates, risorse statiche (css, js, immagini)
  - (es. Layout default) -> Create a new storefront theme | Magento 2 Developer Documentation
- **Templates**
  - hints -> Create a new storefront theme | Magento 2 Developer Documentation
  - (aggiungere) ?templatehints=magento
  - per template hints su inspector: https://github.com/magespecialist/m2-MSP_DevTools
- **Layouts**
  - <current_theme_dir>/<Namespace>_<Module>/layout/
  - <parent_theme(s)_dir>/<Namespace>_<Module>/layout/
  - <module_dir>/view/frontend/layout/
  - <module_dir>/view/base/layout/
- **Styles**
  - Theme styles <current_theme_dir>/web/css/
  - Module theme styles <current_theme_dir>/<Namespace>_<Module>/web/css/
  - Parent theme styles <parent_theme_dir>/web/css/
  - Parent theme Module styles <parent_theme_dir>/<Namespace>_<Module>/web/css/
  - Module styles for the frontend area <module_dir>/view/frontend/web/css/
  - Module styles for the base area <module_dir>/view/base/web/css/

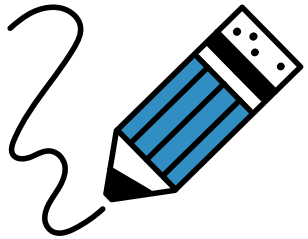


Less, Css e Grunt.



- **Stili Css / Less**
  - Cascading style sheets (CSS) | Magento 2 Developer Documentation
  - less e css
  - @import e @magento_import
  - Css / Less, Merging, Minifying, Critical path
  - Compilazione Less files:
    - server side (production mode) o client side (via browser in developer mode);
  - Grunt
    - Installazione
      - npm install [-g] grunt-cli (omettere l'opzione "global" in caso di errori in container)
      - package.json.sample > package.json
      - Gruntfile.js.sample > Gruntfile.js
      - grunt-config.json.sample > grunt-config.json
      - npm install
      - npm update
      - themes.js into local-themes.js in the dev/tools/grunt/configs/ directory.
      - grunt exec:<theme>, grunt less:<theme>, grunt watch:<theme>
      - Live reload su Warden: LiveReload Setup

Less, Css e Grunt.

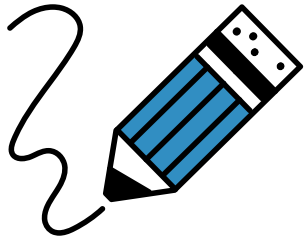


- **Esempio di configurazione su Grunt**
  - Su dev/tools/grunt/configs/local-themes.js

```
nome_tema: {
    area: 'frontend',
    name: 'Vendor/theme',
    locale: 'it_IT',
    files: [
        'css/source/_extend', // * questo esiste realmente nella cartella indicata
        'css/styles-m', // ** questo ereditato da blank, non presente sul tema
        'css/styles-l'  // ** questo ereditato da blank, non presente sul tema
    ],
    dsl: 'less'
}
```

- Comandi comuni:
  - grunt exec:nome_tema && grunt less:nome_tema && grunt watch

Come personalizzare aspetto via css



- **In assenza di grunt:**
  - **rm -rf pub/static && rm -rf var/view_preprocessed/**

- **Le vie più semplici per la personalizzazione via less/css:**
  https://developer.adobe.com/commerce/frontend-core/guide/css/quickstart/customize-styles/

- **Via più semplice per sovrascrivere:**
  https://developer.adobe.com/commerce/frontend-core/guide/css/quickstart/customize-styles/

- **_module.less, per estendere ma anche per sovrascrivere:**
  https://developer.adobe.com/commerce/frontend-core/guide/css/quickstart/customize-styles/

Anatomia base di un modulo

- **Magento2-Module**
  - All'interno di ogni modulo devono essere presenti alcuni elementi;
  - Tutti gli altri sono opzionali e possono essere utilizzati soltanto quando serve.
  - La tipologia di modulo "comune" a cui faremo riferimento è:
    - **magento2-module:**
      - (moduli che aggiungono funzionalità su Magento)
  - Gli elementi obbligatori di un modulo sono:
    - **registration.php**
      - è utilizzato da magento ma anche da composer (determina autoloading e quali file includere automaticamente);
    - **composer.json**
      - contiene le informazioni principali del modulo soprattutto al fine della gestione con composer. Vanno inserite qui eventuali dipendenze;
      - per la nomenclatura è utile seguire quella usata dai moduli magento su vendor/
      - contiene le hard dependencies
    - **etc/module.xml**
      - file fino ad oggi obbligatorio, verrà deprecato e in sostituzione rimarrà unicamente composer.json;
      - contiene il nome e la versione del modulo;
      - contiene un nodo opzionale "sequence" all'interno del quale è possibile definire un elenco di moduli da cui a sua volte dipende (i moduli indicati verranno caricati per primi - v. config.php - e magento ne impedirà la disabilitazione se uno di loro è incluso in questa lista) - soft dependencies;
  - per approfondire il concetto di module dependencies su Magento2: Module dependencies | Magento 2 Developer Documentation

Architettura - Moduli

- **Creiamo il nostro primo modulo**
  - creiamo la cartella app/code/Accenture
  - creiamo la cartella app/code/Accenture/Collections
  - creiamo i files obbligatori:
    - registration.php
      - prendiamo come esempio da vendor/magento/module-catalog/registration.php e cambiamone i parametri
    - composer.json
      - prendiamo come esempio da vendor/magento/module-catalog/composer.json e cambiamone i parametri
    - etc/module.xml
      - prendiamo come esempio da vendor/magento/module-catalog/etc/module.xml e cambiamone i parametri
      - indichiamo come unica dipendenza “Magento_Catalog”.
  - eseguiamo il comando:
    - bin/magento module:enable Accenture_Collections
    - bin/magento setup:upgrade
  - controlliamo il file app/etc/config.php

Cache

L’importanza della cache nello sviluppo: DevDocs

- **pubblica (**server side)
- **privata** (client side)
- **tipi di cache:**
    - DevDocs
- **comandi utili e frequenti:**
    - bin/magento cache:clean [tipo_cache]
        - solo cache abilitate
    - bin/magento cache:flush [tipo_cache]
        - svuota l’intera cache storage
    - bin/magento cache:status [tipo_cache]
    - bin/magento cache:enable [tipo_cache]
    - bin/magento cache:disable [tipo_cache]
- **durante lo sviluppo:**
    - disabilitare:
        - block_html, full_page, layout (se si stanno operando modifiche frequenti e intensive sui layout)

Esercitazione

- **Preparare il proprio progetto**
  - **impostare configurazioni di base**
  - **impostiamo il nostro tema**
  - **creiamo un modulo con**
    - **vendor name “Accenture”**
    - **module name “Collections”**
- **Proposte per moduli**
  -

Giorno 3

ORM in Magento2

Dependency Injection

Attributi Prodotto e Data Patch

Data Patch

- **Le Data Patch:**
  - Integrano automaticamente dei contenuti o delle variazioni ai dati;
  - Sono automaticamente applicate al setup:upgrade;
  - Quelle applicate vengono memorizzate nella tabella “patch_list”;
    - rimuoverle da questa tabella per rieseguirle.

- **Esempi:**
  - cms -> vendor/magento/module-cms/Setup/Patch/Data/CreateDefaultPages.php
  - creazione attributo prodotto -> vendor/magento/module-catalog-url-rewrite/Setup/Patch/Data/CreateUrlAttributes.php
  - modifica attributi prodotti -> vendor/magento/module-configurable-product/Setup/Patch/Data

Orm in Magento2

- **ORM**
- **Persistence layer | Magento 2 Developer Documentation**

  - Ormai affermato l'utilizzo del declarative schema per la gestione del db;
    - aggiungere la generazione del whitelist.json:
      - bin/magento setup:db-declaration:generate-whitelist [options]
  - Model: Oggetto -> \Accenture\..\...\....\CollectionModelFactory
  - Resource Model: gestisce le operazioni;
  - Collection: gruppo di model;
  - Repository: personalizzato, semplifica le operazioni facendo da wrapper per il resource model.
    - https://devdocs.magento.com/guides/v2.4/extension-dev-guide/searching-with-repositories.html

  - Quando usare le factory (Model e Collection) e perchè;
  - DB Schema e i vecchi Setup: Configure declarative schema | Magento 2 Developer Documentation
  - Data & Schema Patch: Develop data and schema patches | Magento 2 Developer Documentation

Dependency Injection

- **Conosciamo la Dependency Injection:** Dependency injection | Magento 2 Developer Documentation
- **Conosciamo l'Object Manager:** ObjectManager | Magento 2 Developer Documentation
- **Operazioni in di.xml (**The di.xml file | Magento 2 Developer Documentation**)**
    - Preferences
        - sostituzione
    - Type
        - attributi
    - Virtual Types
        - "nuove classi"
    - Factory
        - ObjectManager e dintorni (object pool e repository)
    - Interfaces
        - Public interfaces & APIs | Magento 2 Developer Documentation
    - Proxies
        - Proxies | Magento 2 Developer Documentation
- Come "tenere d'occhio" una classe: Dependency - obtain info from cli
    - bin/magento dev:di:info "Magento\Quote\Model\Quote\Item\ToOrderItem"
- Dependency report:
    - https://experienceleague.adobe.com/docs/commerce-operations/configuration-guide/cli/dependency-reports.html?lang=en

Giorno 4

Plugins

Observer

Controller e Routing

Plugins

- **Quando usare un Plugin:**
  - Quando abbiamo la necessità di intervenire sui dati e procedure gestiti dal metodo a cui vogliamo “agganciarci”;
  - Quando il metodo a cui vogliamo agganciarci è un metodo pubblico;
  - Più nel dettaglio non potrò farlo quando ho a che fare con:
    - Final methods
    - Final classes
    - Non-public methods
    - Class methods (such as static methods)
    - __construct
    - Virtual types
    - Objects that are instantiated before Magento\Framework\Interception is bootstrapped
- **Tipi di Plugin:**
  - before
  - after
  - around
- Come ottenere la visuale di chi sta “pluginizzando” la classe? usiamo sempre questo comando già visto per la DI

Observers

- **Quando usare un Observer:**
  - Quando non ho la possibilità di “agganciarmi” ad un metodo pubblico;
  - Quando le mie operazioni, correlate all’evento, non sono dirette a modificare i dati o le attività di quella specifica classe ma, in risposta all’evento, devono operare delle attività parallele.
  - Files interessati per la creazione di observer:
    - etc/[area/]events.xml
    - Magento\Framework\Event\ManagerInterface
    - Magento\Framework\Event\ObserverInterface
  - Gli altri observer possono essere personalizzati (ad es. disabilitati)
  - Ma nel caso dovessimo modificare il comportamento di un altro observer, allora è meglio utilizzare un plugin che lo intercetti.
  - Alcuni eventi sono automaticamente scatenati (legati ad operazioni sul db), con l’aggiunta dell’_eventPrefix della class:
    - Delete before and delete after
    - Save before and save after
    - Save commit after (once the transaction is complete)
    - Load before and load after
    - Clear
  - esempio finale con _eventPrefix=’theme’:
    - theme_delete_before

Controller e Routing

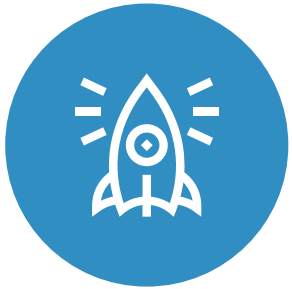
- **Logica dei Controllers e dei Routers**
  - anatomia delle Url:
    - es.: sito.test/catalog/category/view/
      - catalog -> route
      - category -> controller
      - view -> action
      - [parametri opzionali = /chiave/valore/]
  - come individuiamo il “gestore” di una data route:
    - etc/frontend/routes.xml
      - id = utilizzato dai layouts
      - frontName = porzione dell’url
  - Tipologie di “risposte” - View Result e Controller Result :
    - Pagina
    - Json
    - Forward
    - Raw

Giorno 5

Api, Rest e Graphql

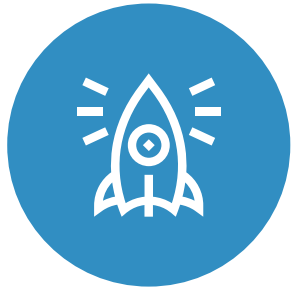
Go Headless

Web Api Rest



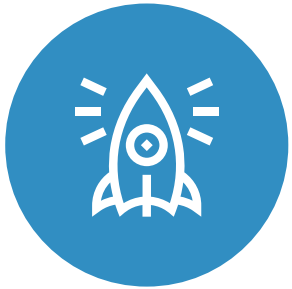
- **Concetti preliminari:**
  - più servizi possono rispondere a una sola chiamata, es:
    - catalogo con chiamata creazione implementa anche stock, immagini e altri servizi;
  - puoi usare per frontend disaccoppiato, per integrare crm, erp o altro
  - quelle REST saranno sempre più destinate alle chiamate backend;
  - per l'accesso con token abilitare un utente System > Permission > All Users > Add New User.
  - in alternativa creare una nuova integrazione da: System > Extensions > Integration > Add New Integration**
    - utilizzare poi client soap o rest per autenticarsi.
      - l'endpoint sarà: https://example.com/rest/default/V1/customerGroups/:id
- **Come iniziare**
  - Guida Web Api su DevDocs
  - Impostiamo il nostro webapi.xml per creare le chiamate custom;
  - Creiamo o personalizziamo la nostra interfaccia;
    - assicuriamoci di utilizzare le giuste acl per le nostre chiamate
  - Creiamo o personalizziamo la nostra dependency injection nel file di.xml;
  - Usiamo uno strumento come Postman per testare le nostre nuove webapi.
    - (utilizziamo come url di partenza (https://app.nomeproject.test/rest/V1/nomepersonalizzato/…)
  - Testiamo i nostri risultati

Graphql



- **Magento2 / Adobe Commerce include già una serie di modelli GraphQL predefiniti**
  e aspira a passare su graphql tutta la parte servita tramite api relativa al frontend (rest rimarrà per il backend)
- **Url di partenza:** https://dominio.ext/graphql
- Ecco alcuni esempi di modelli GraphQL già presenti:
  - ProductInterface: rappresenta le informazioni di base del prodotto, come il nome, il prezzo e la descrizione.
  - CategoryInterface: rappresenta le informazioni di base sulla categoria, come il nome, la descrizione e l'URL.
  - CustomerInterface: rappresenta le informazioni di base del cliente, come il nome, l'indirizzo email e l'indirizzo di fatturazione.
  - OrderInterface: rappresenta le informazioni di base dell'ordine, come il numero dell'ordine, lo stato e l'importo totale.
  - CartInterface: rappresenta le informazioni del carrello, come il numero di elementi nel carrello e il totale.
  - ProductAttributeFilterInput: rappresenta il filtro degli attributi del prodotto.
  - ProductFilterInput: rappresenta il filtro dei prodotti in base a diversi criteri, come l'ID del prodotto, il nome, il prezzo e altro ancora.
  - ProductSortInput: rappresenta il criterio di ordinamento dei prodotti.
  - ProductAttributeInterface: rappresenta le informazioni di base dell'attributo del prodotto, come il nome, il tipo e il valore predefinito.
  - CustomizableOptionInterface: rappresenta le opzioni personalizzabili associate a un prodotto, come il nome e il tipo.
  - e altre ancora…

Graphql



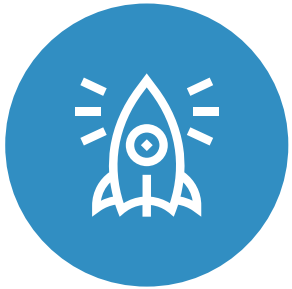
- **Come si definisce una chiamata GraphQl:**
  - schema.graphqls all'interno della cartella /etc del modulo

    (es. modulo di base: vendor/magento/module-graph-ql)
  - product query: vendor/magento/module-catalog-graph-ql/etc/schema.graphqls
  - Esempio più basilare da cui partire come riferimento:
    - vendor/magento/module-customer-downloadable-graph-ql/etc/schema.graphqls
  - Differenze tra query, type e Interface in Graphql:
    - Query: rappresenta una richiesta per ottenere i dati.
    - Type: rappresenta la struttura dei dati che il server GraphQL può restituire.
    - Interface: definisce un insieme di campi che devono essere implementati da un tipo di dato specifico che le implementa
- **Reference**: https://developer.adobe.com/commerce/webapi/graphql/reference/
- **Riferimenti per Aem e Graphql**
  - AEM and Adobe Commerce (Magento) integration Using Commerce Integration Framework
  - Run Graphql queries and mutations | Commerce Web APIs
  - Gestione dei contenuti in AEM Sites | Adobe per le aziende

Graphql

Esempio di chiamata Graphql:

```
{
  products(filter: {sku: {eq: "24-UG05"}}) {
    items {
      name
      sku
      }
  }
}
```



Giorno 6

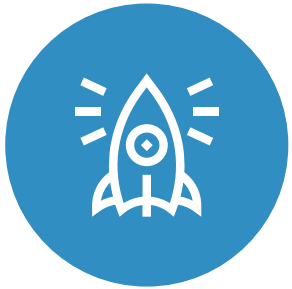
Tecniche di sviluppo ed esercitazioni

Best Practices e Troubleshooting

Adminhtml

- **Opzioni di personalizzazione**
  - Tema: Create an Admin theme | Magento 2 Developer Documentation
  - Admin Design Library: Admin Design Pattern Library | Magento 2 Developer Documentation
  - Admin Style Guide: Admin Style Guide | Magento 2 Developer Documentation
  - Per lo sviluppo in adminhtml, particolare attenzione a:
    - acl.xml
    - adminhtml/menu.xml
    - adminhtml/system.xml *(il più usato)*

Production Mode e Performance



- **Application Modes:**
  - About Magento modes | Magento 2 Developer Documentation
  - Set the Magento mode | Magento 2 Developer Documentation
    - bin/magento deploy:mode:set production

- **Performance Best Practices**
  - **Performance Best Practices | Magento 2 Developer Documentation**
  - Configuration: Configuration best practices | Magento 2 Developer Documentation
  - Servizi: Advanced setup | Magento 2 Developer Documentation
  - Javascript: Advanced JavaScript bundling | Magento 2 Developer Documentation
  - Magepack: magesuite/magepack: Next generation Magento 2 advanced JavaScript bundler.

Easter Eggs

- **Mage2Tv CacheClean:**
  - Installazione: https://github.com/mage2tv/magento-cache-clean
  - Configurazione PhpStorm: https://github.com/davidalger/warden/issues/258#issuecomm
- **Markshust Simpledata**
  - **https://github.com/markshust/magento2-module-simpledata**
- **New Relic**
  - https://docs.newrelic.com/docs/apm/agents/php-agent/frameworks-libraries/magento-spe
- **PhpStorm Plugin:**
  - https://plugins.jetbrains.com/plugin/8024-magento-phpstorm
- **Magerun**
  - https://github.com/netz98/n98-magerun2
- **Want More….?**
  - https://github.com/aleron75/mageres

Grazie!

Domande, richieste?
salvo[at]syntheticlab.it

Twitter:
https://twitter.com/scapritta

Linkedin:
https://www.linkedin.com/in/salvatorecapritta/