42CMS

Introduction

A business group wants to expand their brick and mortar retail stores online, but the company's business users have no dev skills! They want you to create a Content Management System to easily handle their e-commerce needs and to generate their storefront e-commerce websites.

Objectives Overview

The objective of this project is to respond to **general** and **specific** needs of business users. Your project should respond to the following requirements:

- **CMS**:
  - **Web API**: An api to access and update our e-commerce database. This web API is used by the back-office website. With this api, we should be able to create business accounts, create products and sort them into catalogs, customize business object types, and export and import catalogs.
  - **Back Office Website**: A front-end web application for us to manage the e-commerce database. This uses the aforementioned web API.
- **Storefront(s)**: Drawing from the database and back-end configurations, the CMS should dynamically create front-end e-commerce websites.

General Instructions

You are free to use whatever tech and languages you wish. You cannot use libraries which specifically handle required project functionalities.

Required Functionalities

Products, Catalogs and Categories

We want to be able to:

- create and edit at least three types of products: basic products, master products and variant products:
  - **Master Product**: This is an abstract product meant to represent a group of products that differ by one or more **variation attributes**. For example, a T-Shirt with two variation attributes, size and color, can be considered a Master Product.
  - **Variation Product**: This product is a concrete product which varies according to a Master Product's variation attributes. For example, a perfume can come in different bottle sizes, such as 30ml and 60ml (each one of those is a variant). A Variant Product must have its Master Product's variation attributes defined. A Variant Product cannot exist without a Master Product. A Variation Product's attributes can override its Master Product's attributes, but otherwise inherits them.
  - **Basic Product**: This is a simple, standalone, concrete product with no relation to a Master or

Variant Product.

- have at least the following product attributes: (see *Object Type Definitions*)
    - ID or SKU (string)
    - EAN (string)
    - name (string, localizable)
    - description (string, localizable)
    - images (image array, site-specific)
    - primary image (image, site-specific)
    - A product's default image
    - online (boolean, site-specific)
    - Determines if a product is online
    - onlineFrom (date, site-specific)
    - Date from which the product should be online
    - onlineTo (date, site-specific)
    - Date after which the product should be offline
    - available (boolean, site-specific)
    - Current availability based off of inventory
- create and edit catalogs.
- create and edit categories.
- assign our products to one or more catalogs.
    - **Catalog**: A collection of products and categories. A catalog can be assigned to one or more sites.
- assign products to one or more categories within each catalog.
    - **Category**: A group of products within a catalog. Categories can be nested to hierarchically organize products. Products should have a primary category for each catalog they are assigned to. For example, Men > Shoes > Running > *My Shoe* is the primary category path in which *My Shoe* is found, but it can also be found on the category page: New Products > Men > *My Shoe*

Extensible Objects and Object Type Definitions

A **Business Object** represents any back-end entity, such as a Master Product, a Catalog, or a Category.

We want at least the following Business Objects to be extensible:

- Products
- Catalogs
- Categories
- Sites

An **Object Type Definition** represents a Business Object template (like a class) with all its attributes and attributes types defined. For each attribute, the Object Type Definition should tell us:

- if it's a system or custom attribute
- if it's localizable, site-specific, or neither
    - **localizable**: the attribute value is different if viewed on the storefront in different locales (see *Locales*)
    - **site-specific**: the attribute value is different if viewed on different sites (see *Sites*)

    - neither: the value is always the same
- the attribute type (see below)

We want to be able to view and edit Object Type Definitions.

We want to be able to:

- extend Extensible Objects with custom attributes. We want the following types:
    - string
    - number
    - boolean
    - image
    - Date
    - array of one of the previous types
    - example: 'ingredients' is a product attribute which is an array of strings of ingredient names
    - enum of one of the previous types
    - example: 'tag' is a product attribute which can be one of the following values: 'new', 'Valentine's Day' or 'Mother's Day'
- choose if these attributes are localizable
    - for example, we want to display a category or product name differently for users viewing a page in French than for those viewing the page in English
- or choose if these attributes are site-specific
    - for example, we want a product's primary image to be different for our Japanese site than for our EU site.

Locales

Many business object attributes will be visible on our storefront website, such as product names and descriptions. We want to tailor the site's display language to that of the user.

We want to be able to:

- define locales (language and country)
- manage allowed locales globally and per site
- define attribute values by locale (see *Object Type Definition*)
- set up a system of locale fallbacks. For example, we would like to set the fallback of the "en_CA" locale to "en_US", so that if we've defined a product name in "en_US", but not in "en_CA" and are accessing a site in "en_CA", the product name under en_US will be displayed.

Sites

Sites are used to handle different customer bases or markets.

For example, we'll want a site or the European market, one for the North American market and one for the Chinese market.

We want to be able to:

- set the url for the site (For this project, you can simulate the use of a url in any way you see fit)
- assign Catalogs to a Site
- assign Locales to a Site
    - customers should be able to toggle between those assigned locales on the storefront
- assign a Price Book to a Site
- activate and deactivate a Site

Business Users and Permissions

A business user can access and modify products, catalogs etc through the back-end interface.

We want to be able to:

- have an admin who can create users
- manage permissions by user for at least the following actions:
    - editing products, catalogs and categories
    - managing extensible object attributes
    - (Bonus) importing and exporting xmls (see *Import Export System*)

Image Management

We want to be able to:

- upload, download and delete images
- assign an image or images to products
- set a primary image to a product

Price Books

Price Books contain information about a set of products in a currency

We want to be able to:

- create price books
- set the currency to a price book
- assign products to a price book
- set the price for products in a price book

Storefront

We want the following pages:

- home page
- category page
- product page
- checkout page
- cart page

Defense Session

Prepare to run unit tests to demonstrate the features you implemented.

Bonus Functionalities:

Import and Export System

We want a simple system to import and export data to and from the data-base in an xml format. We want to be able to:

- import and export Products
- import and export Catalogs and Categories
- import and export Object Type Definitions

Templating System

We want to assign an html template as an attribute to a product in order to display that product with that template. For example, we want to use a different template for a product viewed on our French website, and a different template used on the Japanese website (Create 'template' as a localizable or site-specific attribute for products).

Bonus Product Types

We want to be able to create additional product types:

- **Product Bundles**: A group of products that can only be sold together for a single fixed price.
- **Product Sets**: A group of products that appear together on the storefront and can be purchased together or separately. A set does not have its own price; its price is simply the sum of the set's products.

Other Bonus Ideas:

- Inventory and sales management system
- Customer tracking system (CRM)
- ADA or WAI Accessibility