CORESHOP
SOLUTIONS FOR PIMCORE

Technical Discussion:

PIMCORE DATA MODELING AND INHERITANCE

A data model pertains to the data, its characteristics as well as how it's related to other data. It is also required to understand how to address the needs, and how to design the database for your organization. At Pimcore, we make sure data modeling stays as flexible as possible, since this enables a PIM and MDM to grow with your organization. Pimcore includes a web-based data modeling engine to create a new product data model within minutes.

Pimcore Data Modeling Key Features:

- Flexible Data Modeling: Flexible data modeling is key to Pimcore PIM/MDM. It includes a web-based data modeling engine to create a new product data model within minutes. More than 40 high-performance data types capable of handling millions of products or other master data with thousands of attributes each. Perfect both for operational PIM/MDM and analytical PIM/MDM.
- Multi-Lingual Data Management: Pimcore PIM/MDM works with multiple languages – any number of languages. It provides the flexible data model, data editing, and translation interfaces for effectively working with product data within an international or global context. The Pimcore administration interface is available in 20 different languages, including English, German, French, Italian, Spanish and Chinese.
- Object Relations: A flat data model (as in Excel) is rarely seen in real-life enterprise product information management. To put real value into product data, its data model must constantly represent the full range of attributes, relations and specifications and adapt in real-time to changes and additions. Pimcore provides object relations and advanced concepts of field collections and objects bricks to deliver a flexible and scalable data model during the entire product lifecycle. Object relations are used for product Categories, Features, Technologies and Collections. They make your data model scalable.

- Classification Standards: Handling text, media, static attributes, and relationships is just the beginning of data modeling with Pimcore. The next logical step is the classification and structure assignment of product data based on industry specific classification systems such as eCl@ss, ETIM or GS1. The Pimcore classification store is an advanced data modeling component to manage any type of classification systems.
- Digital Asset Management: Product Information Management is not only text and attributes, but also unstructured data such as media assets. The Pimcore Data Manager seamlessly integrates the full-featured Digital Asset Management Pimcore DAM to manage, integrate and relate images, graphics, documents, audio and video. Pimcore DAM manages, integrates and relates images, graphics, documents, audio and video.
- Data Inheritance: The concept of data inheritance is a key capability to Pimcore data modeling. Data inheritance enables the inheritance of product data to subordinate groups within the tree. This concept reduces the time-to-marke and simplifies all editorial processes, by maintaining data just one time at one place.

Pimcore Objects Discussion

Objects are the PIM part of Pimcore and are the way to manage structured data within Pimcore. Based on a class definition that defines structure and attributes, objects can be used for any structured data, such as: products, categories, persons, customers, news, orders, blog entries, For the attributes many datatypes (simple ones and complex ones) are available.

Pimcore objects are literally objects in the sense of object-oriented programming. The class definition can be defined through a user-friendly graphical user interface (GUI), but nevertheless in the background a plain PHP class is created, which can profit from inheritance and can be enhanced and accessed with custom PHP coding. Data objects can be instantiated and filled within Pimcore's backend or within your custom code by using the PHP API and using common programming paradigms (create new instances, using getter and setter, etc.). To save objects, call the " save: method and Pimcore takes care of the rest. So, it is also easy to serve objects from external systems like CRM, ERP, PIM or asset management systems.

Pimcore Data Inheritance and Parent Class Discussion

A very important feature in connection with PIM is data inheritance. Data inheritance means, that objects of the same class can inherit data from their parent objects in the object tree. Pimcore provides two types of inheritance. While data inheritance allows the inheritance of object data from a Parent Class from a tree hierarchy of objects, a developer can also modify the PHP class hierarchy with the parent class settings of object classes. A typical use case is the storage of product data. An example would be to define a group of products which have

many attributes in common and that they differ in just a few or unlimited number of attributes (for example size, color, etc.). First, create a Parent Product which stores all the common attributes, then add Child Products and stipulate attributes in which the products may differ (size, color, etc.,). All other attributes are inherited from the common parent product.

Pimcore Object Variants Discussion

The best way to explain the use and function of Object Variants is through a use case: The objective is to store all product related data in Pimcore as the single source of truth. Many of these products are variants of each other, for example T-shirts: most t-shirts' attributes have the same values and they just differ in color and EAN code such as a yellow t-shirt, a blue t shirt, a red tshirt etc. One way to store this is to make a generic t-shirt object and then create for each variant a child object within the tree. This approach works fine, but if you have dozens or even hundreds of variants, your object tree becomes quite big and confusing. This is where object variants come in. Basically, they are just objects which aren't shown in the object tree. In the tree, you just create the generic t-shirt. For each variant of this t-shirt, you create an object variant, which is not shown in the object tree but in an own tab within the object editor. There is no other difference between objects and variants, therefore variants support all functionalities as objects do. You can create hundreds of object variants without impairing your object tree.

Pimcore Object Classes Discussion

To implement Pimcore objects, classes must be defined. Defining a class consists of two parts: Define the attributes of the object and then define the layout for the object editor. Layout object properties can be grouped into panels, which incorporate the layout areas north, east, west, south and center and additionally they can be positioned into tab panels. This allows logical structuring of object attributes into smaller units of data belonging together. It depends on the use case for how data should be grouped and structured. Common applications are tabs/groups for different languages or logical groups like basic data, media, sales data, etc. In addition to the master editor layout, Custom Layouts for different views on the object data can be defined.

Class attributes are defined from a set of predefined data types. These data types define not only the type of data such as text, number, image, reference to another object etc. but also how data input can be achieved and how data is accessed. Examples of data types are: text, number, date, select, relational, structured, block, geographic, CRM and other.

CORESHOP
SOLUTIONS FOR PIMCORE

Summary: It's all about Data Management

Whether it's managing the business process or gathering business intelligence, efficient enterprise data management is a prerequisite. With a deep focus on user-friendly and consistent organization, aggregation, classification, and translation of rich product information, Pimcore is uniquely placed to address PIM and MDM needs of enterprises. We recognize that efficiency and accuracy in handling data is at the core of a reliable data governance. Besides, being an open source data management platform, Pimcore provides ample scope for scalability, consistency, and optimization.