COREHOP
SOLUTIONS FOR COMMERCE

PIMCORE 6 Best Practices Guide

Pimcore 6: Is the latest and most powerful version of Pimcore.

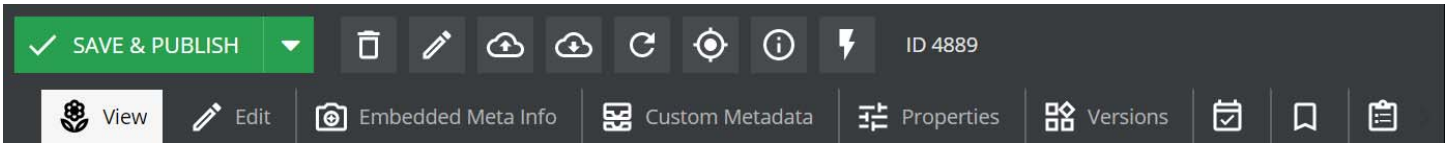
It is made on Symfony 4 framework. Currently, Symfony 4 is the fastest PHP framework available. Symfony 4 integrates seamlessly with Symfony Flex to automate the most common tasks performed on applications. Pimcore is based on Symfony 4 and can use all advantages of Symfony 4.

Technology stack uses in PimCore 6

Developers and Community of Pimcore have put in a great effort into standard and optimize the code. This not only makes code lighter, transparent but also makes it easy for community developers to contribute. Pimcore 6 gave a new look to its admin panel. It gives a totally new experience to PimCore admins and editors. PimCore is a fully open sourced product and available at zero license cost for developers, agencies, and enterprises.

The following are highlights of some of admin panel UI/UX changes

It is made on Symfony 4 framework. Currently, Symfony 4 is the fastest PHP framework available. Symfony 4 integrates seamlessly with Symfony Flex to automate the most common tasks performed on applications. Pimcore is based on Symfony 4 and can use all advantages of Symfony 4.



Pimcore introduced a new collection of icons that are matched with the flat design approach. Now they are simple, clear and easy to spot.

WCM shortcut

Content Management system is now called Web content management system. There is lot of clutter which is removed from WCM.
New admin panel all in all is cleaner, less clustered and easier to use.

Full range of UTF-8 characters

Use of UTF-8 Character set enables PimCore 6 for non-Latin installations. Pimcore 6 now becomes compatible for Chinese and logographic language installations.

Vertical tabbing

Pimcore has improved the compatibility of 'data objects' by adding support for adjustable tab positions. Now 'Tab panels' and 'localized fields' can be displayed on the top, bottom, left or right.

CORESHOP
SOLUTIONS FOR COMMERCE

Document Management & WMS

- Use Pimcore Document for displaying any content on the CMS frontend. Don't create custom PHP pages unless necessary as you will not be able to get all benefits of Pimcore.
- Design folder structure and hierarchy of Documents to match the sites structure and navigation needs.
- Carefully identify Layout Blocks and Elements based on CMS needs to make best use of Pimcore Document Types and Editables. e.g.

  Newsletter documents are the way to create and send newsletters directly within Pimcore. Similarly, Snippet makes it easier to extract often used contents into reusable containers. Can be embedded in pages or nested into other snippets.

- When creating documents correct type of documents based on functionality specific for the intended use-case.
- Use editables of appropriate type based on type of information to be displayed, repetition of element blocks and control provided to editor.
- When creating a custom brick provide a brick ID which is to be unique throughout the system.
- Please make sure your brick is defined inside a bundle as otherwise your templates can't be auto-discovered.
- As far as possible use Pimcore standard navigation implementation since it builds a navigation container based on the existing document structure.

COREHOP
SOLUTIONS FOR COMMERCE

Document Management & WMS

- Make sure Pages and links have Navigation Settings properly defined else they may not appear in the navigation.
- To benefit from the cache Don't use Pimcore\Model\Document objects directly in the navigation templates / partial scripts, because this would result in loading all the documents again in the navigation and bypasses the caching mechanism of the navigation container.
- Make sure that the documents and its parent documents are published and that the document itself as well as all its parents have a navigation name set.

Asset Management

- Organize assets – documents, images and videos - into proper structure based on website structure and user needs.
- Set role-based permissions at folder level to ensure correct access to authorized users.
- Make sure all external libraries required by Pimcore are properly setup to ensure all features work optimally. E.g. imagemagik, ffmpeg, color profiles etc.
- Identify dimensions and formats of image thumbnails needed and configure automatic thumbnail generation accordingly so that Pimcore can calculate and provide optimized images for different channels.
- If using image transformations, note that the transformations are performed in the order from the top to the bottom and incorrect order may result in undesired results.

CORESHOP
SOLUTIONS FOR COMMERCE

Data Objects

- Define the Data Objects and their relations to reflect entities involved in the actual business process as far as possible.
- When defining data object layout, consider various user groups and use cases to understand 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.
- Make sure the class name is a valid PHP class name. Note that renaming a field means the loss of data from the field in all objects using this class.
- Make sure appropriate fields are indexed, made searchable to allow efficient search.
- When defining class attributes carefully consider data type, data range, validations, structured or unstructured, reference to other objects etc. It is not recommended to add custom validations later at view level. Also define appropriate object configuration - mandatory, not editable, invisible,
- Use structured data types – Collections, Bricks etc. to implement complex data structures.
- Identify attributes that need localization support and define attributes accordingly. Please note that moving a field from outside (normal object field) into the localized field container means the loss of data from the field in all objects using this class.
- Use the layout settings to apply custom CSS to any object field. Avoid applying CSS class on attributes at view level.
- For reference field in the object field definition correctly configure which types and subtypes of elements are allowed.

CORESHOP
SOLUTIONS FOR COMMERCE

Data Objects

- Use Getters and Setters to fetch / assign attribute values. Direct assignment may result in undesirable behavior. E.g.

  $object->multihref = null; will not work to clear the list of elements in the multihref when lazy loading a list.

- To make benefit from Pimcore caching feature use Lazy Loading wherever possible.

- Don't use Block if you are planning to query the data. The block data basically just gets serialized into a single database column hence query on elements will not work.

- Use customize layouts and views to enable different views of object to different user groups. Avoid adding customization at view level to achieve this end.

- Don't use Perspectives and Custom View to restrict access to data. These are not intended to be used to restrict access to data.

COREshop
SOLUTIONS FOR COMMERCE

General - Coding Conventions, Folder Structure & Configuration

- ✔ Try to follow and implement PHP-FIG PSR standards below in development with Pimcore.
  - PSR-1 (Basic Coding Standard)
  - PSR-2 (Coding Style Guide)
  - PSR-3 (Logger Interface)
  - PSR-4 (Autoloading Standard)
  - PSR-6 (Caching)
- ✔ Adhere to development only in recommended folders based on the kind of development, without modifying core Pimcore files.
  - /pimcore/ Core files of Pimcore, do not change anything here.
  - /var/ Private generated files - not accessible via the web (cache, logs, etc.). These should not be added to repository.
  - /vendor/ All third-party libraries are there. It's the default location for packages installed by Composer
  - /web/ This is the document root (public folder) for your project - point your vhost to this directory!
- ✔ For custom development, make sure your custom code goes into following locations within the total architecture:
  - Apps/website within the MVC component: All the solution specific implementations like all controllers, views, models for your website.
  - Plugins/Bundles, custom modules: All implementations and modules you might want to reuse with other solutions.

CORESHOP
SOLUTIONS FOR COMMERCE

General - Coding Conventions, Folder Structure & Configuration

- Do not alter Pimcore constants defined in /pimcore/config/constants.php. Either create a file in /app/constants.php or define an environment variable named after the constant or define an environment variable in a /.env

- Any code to designed to influence Pimcore's startup process should be done by adding a file in /app/startup.php. Core bootstrap files should not be modified.

- It is recommended to use composer to add custom modules/bundles and manage other Pimcore dependencies.

- Define new Controllers by extending the abstract Frontend Controller class provided by Pimcore (Pimcore\Controller\FrontendController) to make use of additional methods specific to Pimcore.

- Before writing any custom utility method for view scripts check if it is provided by Pimcore Helper Classes or Standard Symphony Helper classes.

- For any custom routing / redirection, try to use the features of Pimcore Documents instead of writing custom code or changes .htaccess.

- Keep in mind Priority of Routing processing by Pimcore. These routes are processed in a specific priority order as described below.

    - System / Symfony Routes:
    - Redirects with Priority
    - Pimcore Documents and Pretty URLs
    - Static Routes / Custom Routes
    - Redirects