Fundamentals

Fundamentals

PrestaShop was conceived so that third-party modules could easily build upon its foundations, making it an extremely customizable e-commerce software.

PrestaShop's customization is based on three possibilities:

- Themes,
- Modules,
- Overriding.

Themes are explored in full in the Design Guide: http://doc.prestashop.com/display/PS15/Designer+Guide.
Modules and the override system are explored in this Developer Guide, starting with the "Concepts" section below. You can learn more about each in the following chapters:

- http://doc.prestashop.com/display/PS15/Creating+a+PrestaShop+module
- http://doc.prestashop.com/display/PS15/Overriding+default+behaviors

By default, PrestaShop is provided with more than 100 modules, enabling you to launch your online business quickly and for free.

More than 2000 modules are also available at the official add-ons site.
These additional modules were built by the PrestaShop company or members of the PrestaShop community, and are sold at affordable prices.
As a developer, you can also share your modules on this site, and receive 70% of the amounts associated with the sale of your creations. Sign up now!

Concepts

You should be familiar with PHP and Object-Oriented Programming before attempting to write your own module.

A module is an extension to PrestaShop that enables any developer to add the following:

- Provide additional functionality to PrestaShop.
- View additional items on the site (product selection, etc.).
- Communicate with other e-commerce services (buying guides, payment platforms, logistics, etc.).
- etc.

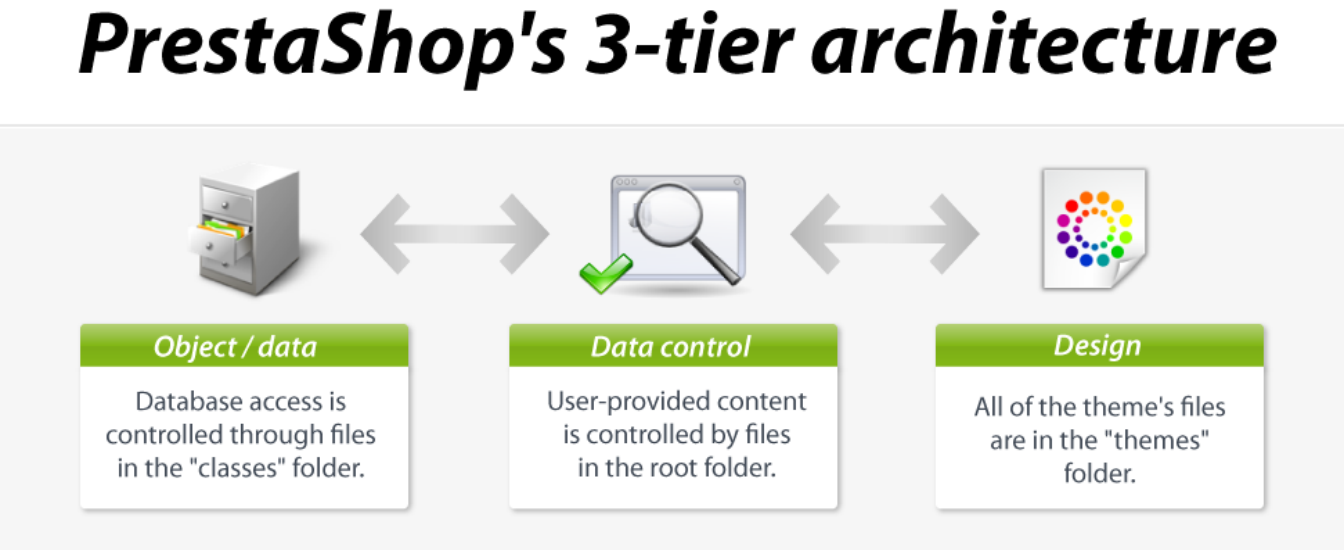
Overriding is a system in itself. PrestaShop uses completely object-oriented code. One of the advantages of this is that, with the right code architecture, you can easily replace or extend parts of the core code with your own custom code, without having to touch the core code. Your code thus overrides the core code, making PrestaShop behave as you prefer it to.

PrestaShop's technical architecture

PrestaShop is based on a 3-tier architecture:

- **Object/data**. Database access is controlled through files in the "classes" folder.
- **Data control**. User-provided content is controlled by files in the root folder.

- **Design**. All of the theme's files are in the "themes" folder.



This is the same principle as the Model–View–Controller (MVC) architecture, only in a simpler and more accessible way.

Our developer team chose not to use a PHP framework, such as Zend Framework, Symfony or CakePHP, so as to allow for better readability, and thus faster editing.
This also makes for better performances, since the software is only made of the lines of code it requires, and does not contain a bunch of supplemental generic libraries.

A 3-tier architecture has many advantages:

- It's easier to read the software's code.
- Developers can add and edit code faster.
- Graphic designer and HTML integrators can work with the confines of the `/themes` folder without having to understand or even read a single line of PHP code.
- Developers can work on additional data and modules that the HTML integrators can make use of.

Model

A model represents the application's behavior: data processing, database interaction, etc.

It describes or contains the data that have been processed by the application. It manages this data and guarantees its integrity.

View

A view is the interface with which the user interacts.

Its first role is to display the data that is been provided by the model. Its second role is to handle all the actions from the user (mouse click, element selection, buttons, etc.), and send these events to the controller.

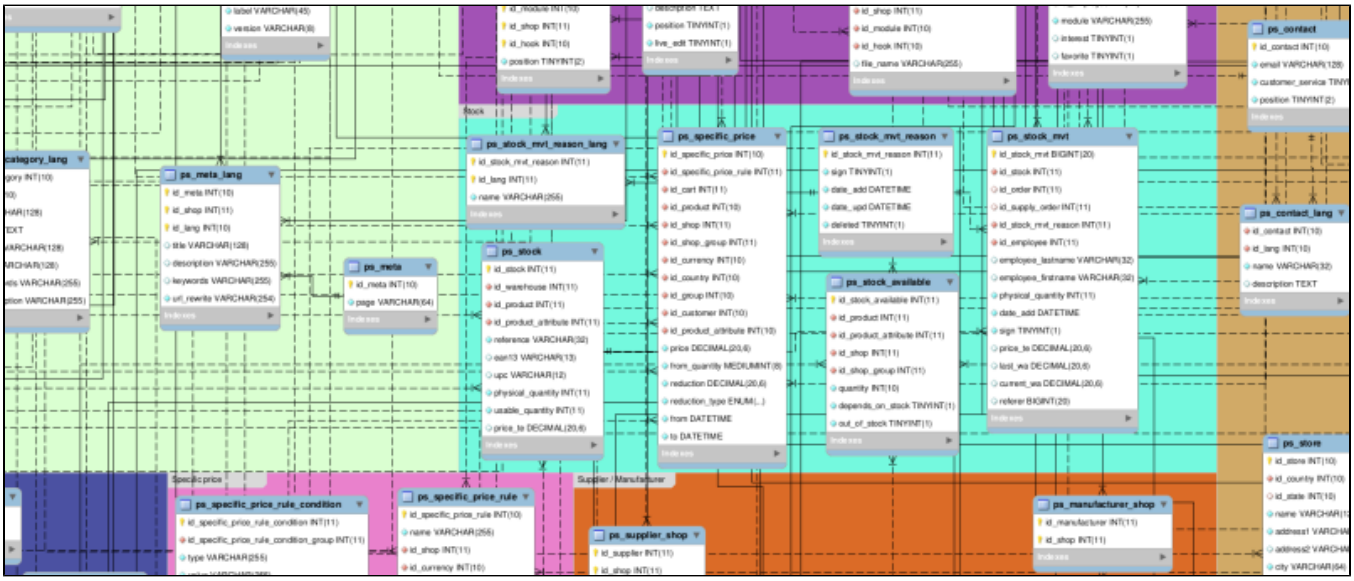
The view does not do any processing; it only displays the result of the processing performed by the model, and interacts with the user.

Controller

The Controller manages synchronization events between the Model and the View, and updates both as needed. It receives all the user events and triggers the actions to perform.

If an action needs data to be changed, the Controller will "ask" the Model to change the data, and in turn the Model will notify the View that the data has been changed, so that the View can update itself.

Database schema



You can download the PrestaShop 1.5 SQL schema in PDF form (3.76 Mb), or in the original MySQL Workbench file format (you will need MySQL Workbench to view it).