

---

## Explication Pimcore – Zend :

---

Avant toute chose, voici les divers outils utilisés au sein de l'entreprise.

**Git** qui est un logiciel de gestion de versions décentralisé. Il permet de travailler en groupe et obtenir les différentes versions qui ont été faite par les membres du groupe. Ainsi que fusionner les différentes versions préalablement déposées.

**Redmine** est une application web libre de gestion de projets (développée en Ruby). Ses fonctions sont les suivantes : la gestion multi-projet, c'est-à-dire la prise en charge de plusieurs projets à la fois, ainsi que la saisie du temps passé sur chacun d'entre eux.

**Zend (version 2)** est un framework qui permet d'accélérer le développement de sites web, faciliter la maintenance et coder de manière « unique » en PHP. Grâce à lui, il est possible de séparer le code en une architecture MVC (Modèle – Vue – Contrôleur) et de permettre l'authentification ainsi que les ACL (droits d'accès).

**Pimcore** qui est un système de gestion de contenu (CMS). Il permet la création et la gestion de sites et applications web (licence BSD : libre). Il est basé sur deux frameworks, Zend (serveur) et ExtJS (client). Les fonctionnalités de ce dernier sont : personnaliser le Back-Office (partie non visible par l'utilisateur) ainsi que l'automatisation de certaines tâches (conversion document office, prévisualisation des médias)

---

## Contexte :

---

Dans ce document nous allons voir comment coder pas à pas une liste de voitures stockées dans une base de données grâce à Pimcore et Zend.

Pour installer Pimcore, il m'a juste fallu me rendre sur leur site (accessible à cette [adresse](#)) et télécharger un dossier Zip, qu'il a simplement fallu décompresser. A partir de là, j'ai pu accéder à la page d'accueil par défaut, puis je me suis connectée dans Pimcore en tant qu'admin afin de commencer le projet.

Pour ce dernier, j'ai utilisé la version 2 de Zend et la version 2.3.0 pour Pimcore.

---

## Développement :

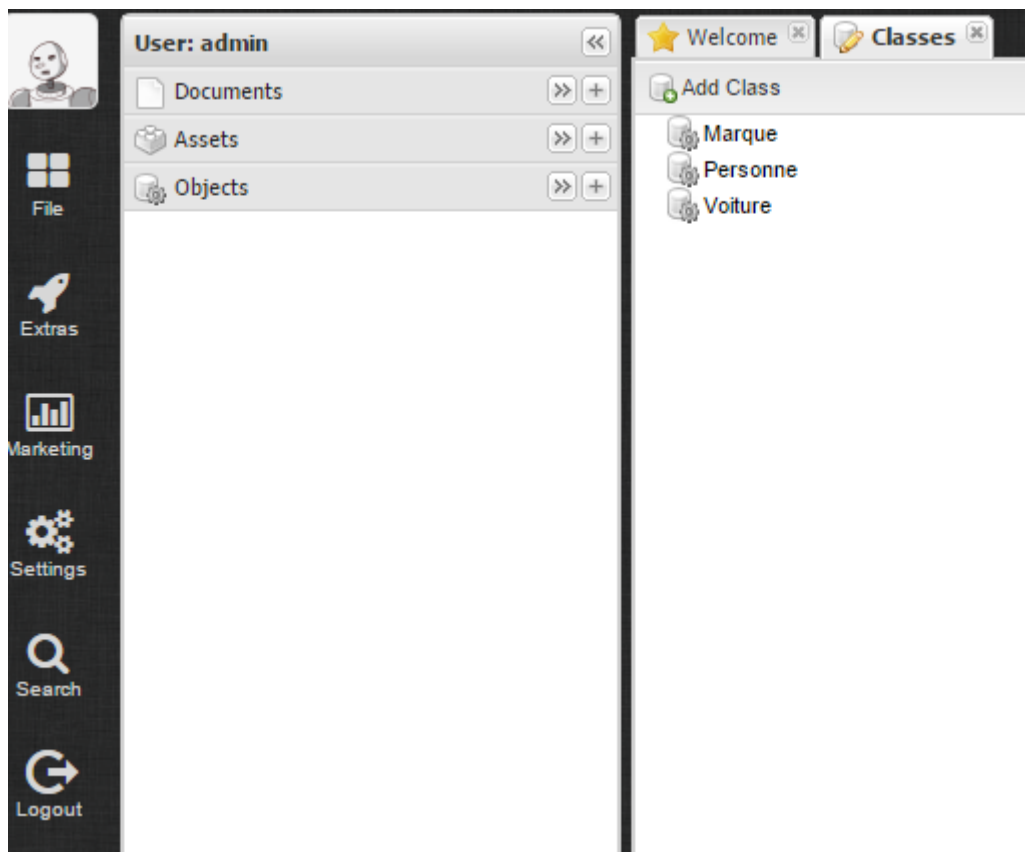
---

➤ Étape 1 :

Les rubriques Documents, Assets et Objects sont les trois grandes parties du CMS Pimcore.

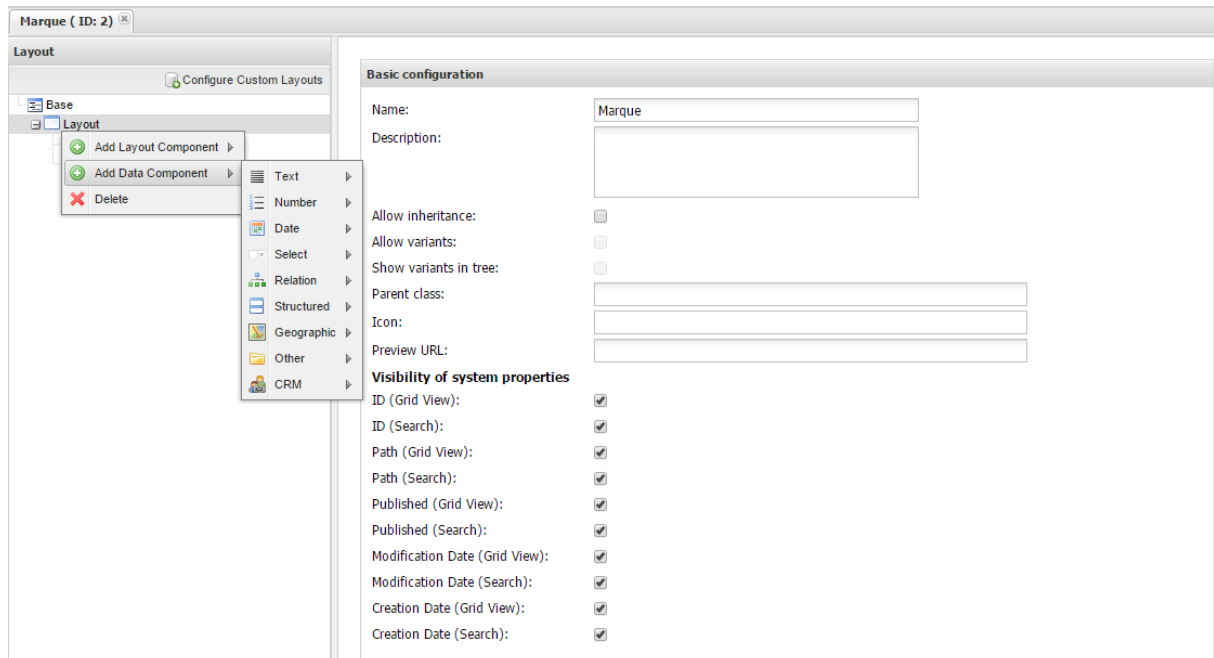
Documents regroupe les fichiers créés, Assets est un regroupement des divers médias que l'on a pu ajouter et enfin, Objects définit tous les objets (poo) qui ont été créés.

A droite de l'écran se situe les classes qui ont été créées (ici Marque, Personne et Voiture). Pour en ajouter une, il suffit de cliquer sur Add Class et de lui donner un nom.



➤ Étape 2 :

Une fois la classe créée, cet écran apparaît. Nous avons donc le choix pour créer soit un champ texte (text), une liste déroulante (select), une date (date) etc. Lorsque l'on a choisi ce que l'on désire ajouter, il suffit juste de préciser le nom du champ (name) et la classe parent s'il y en a une.



➤ Étape 3 :

Ici la classe Marque contient deux champs textes, le nom de la marque et le pays d'où elle est issue. La case « Mandatory field » indique qu'en la cochant le champ devient alors obligatoire. Il est également possible d'ajouter du CSS.

Configure Custom Layouts

Base

Layout

nom

pays

### General Settings (Input)

Name:

Title (Label):

Tooltip:

Mandatory field: ☒

Not editable: ☐

Invisible: ☐

Visible in Grid View: ☐

Visible in Search Result: ☐

Indexed: ☐

### Layout Settings (Pimcore Admin)

CSS Style (float: left; margin:10px; ...):

### Specific Settings (Input)

Width:

Columnlength:

Regular Expression Validation

RegEx:

⚠ RegEx without delimiters, has to work in both JS and PHP (Delimiter: #)

Test string:

Reload Definition Import Export Save

➤ Étape 4 :

Ici nous avons la classe voiture, la classe principale qui contient toutes les informations concernant les voitures (le nom, la marque, la date de mise en circulation, la description, les options, l'image et enfin les personnes qui sont dedans).

The screenshot displays a software development environment with three tabs at the top: 'Marque (ID: 2)', 'Personne (ID: 3)', and 'Voiture (ID: 1)'. On the left, a sidebar shows a tree view under 'Base' with 'Layout' expanded, listing properties: 'nom', 'marque', 'miseEnCirculation', 'description', 'options', 'image', and 'personnes'. The main area is divided into two panels. The left panel, titled 'Layout', contains a 'Configure Custom Layouts' button. The right panel, titled 'Basic configuration', contains the following fields and options:

- Name: Voiture
- Description: (empty text box)
- Allow inheritance: ☐
- Allow variants: ☐
- Show variants in tree: ☐
- Parent class: (empty text box)
- Icon: (empty text box)
- Preview URL: (empty text box)
- Visibility of system properties**
- ID (Grid View): ☒
- ID (Search): ☒
- Path (Grid View): ☒
- Path (Search): ☒
- Published (Grid View): ☒
- Published (Search): ☒
- Modification Date (Grid View): ☒
- Modification Date (Search): ☒
- Creation Date (Grid View): ☒
- Creation Date (Search): ☒

### ➤ Étape 5 :

Voici les différents objets voiture qui ont été créés grâce aux propriétés sur l'écran précédent (étape 4). Nous choisissons donc le nom de l'objet que l'on vient de créer, apparaît ensuite cette page avec les champs déjà défini précédemment.

Il suffit de remplir le nom de la voiture, sa marque (en ajoutant au fur et à mesure grâce au bouton vert situé à droite), la date de mise en circulation, la description, les options disponibles, concernant l'image, il suffit de la télécharger et de l'inclure directement dans l'objet ci-dessous. Toutes les images seront répertoriées dans la partie Assets. Et pour les personnes situées dans la voiture, il suffit de les créer aussi une par une grâce à la classe Personne.

Users: admin

Documents

Assets

Objects

- Home
- marque
- personne
- voiture
  - v3
  - v4
  - v5
  - v6

Welcome | Classes | voiture | v3 | v4 | v5 | v6

Save & Publish | Unpublish | Delete | Reload | Show in Tree | Info | ID 10 | Viture

Edit | Properties | Versions | Schedule | Dependencies | Notes & Events | Children Grid

Nom

Nom: Scenic

Marque

ID	Path	Type				
13	/marque/renault	Marque				

miseEnCirculation: 12/11/2014

description

Moteur essence, ESP, ABS.

Options: Sieges chauffants, GPS

Image

Personnes

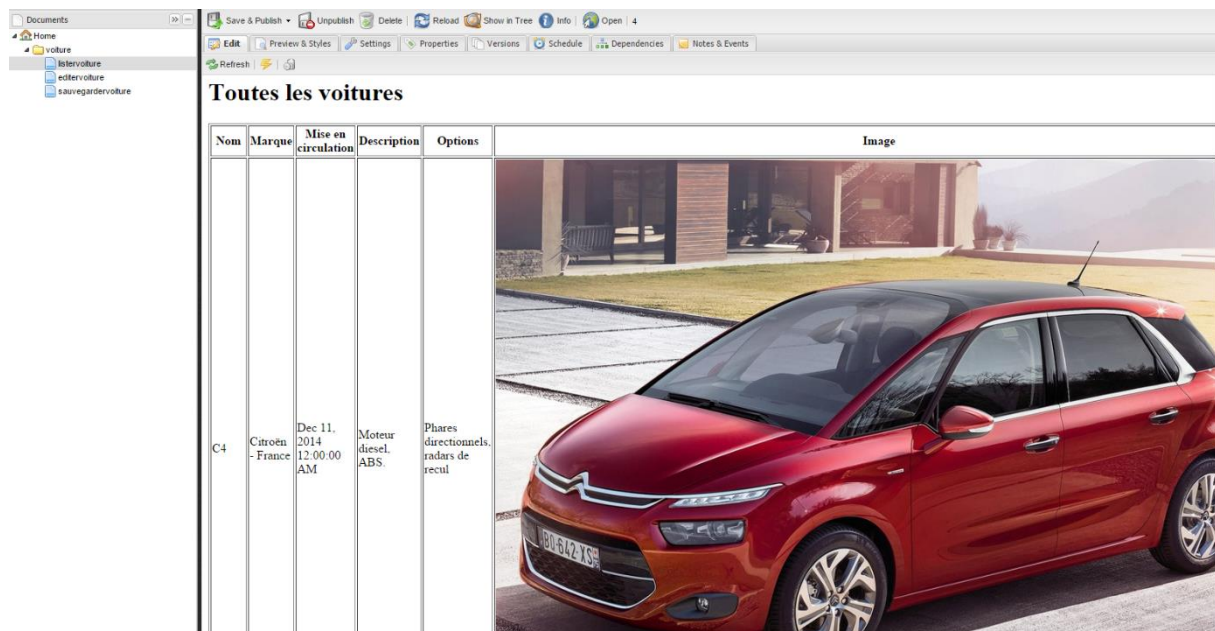
ID	Path	Type				
17	/personne/jacques	Personne				
16	/personne/jean	Personne				
18	/personne/jule	Personne				

➤ Étape 6 :


Nous arrivons désormais aux vues qui faut falloir créer. Dans la partie « Documents » on va créer un dossier Voiture qui représente la classe, et les divers documents « listervoiture », « editervoiture » et « sauvegardervoiture » qui réprésent les vues correspondantes au développement effectué en parallèle dans NetBeans grâce à Zend. C'est-à-dire que dans Pimcore nous rajoutons seulement un champ input, image, select ou autre, mais dans NetBeans ce sera en code PHP par exemple pour rajouter une image le code sera le suivant :

```
< ?php $this->input('content') ; ?>
```

Le nom que l'on met entre parenthèse est celui qui sera utilisé dans la base de données. En revanche, sur une page, il ne peut pas y avoir deux fois le même nom, même s'il y a plusieurs input (dans notre exemple). Il faut donc le renommer explicitement.

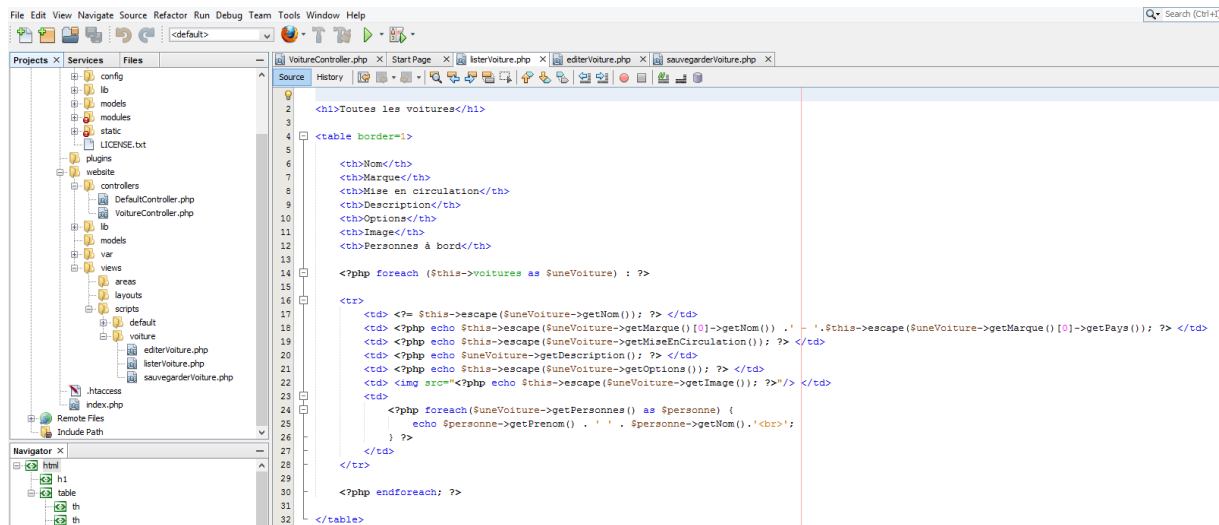


The screenshot shows a web application interface with a sidebar on the left containing a file tree with folders like 'votre' and files like 'listervoiture', 'editervoiture', and 'sauvegardervoiture'. The main content area is titled 'Toutes les voitures' and contains a table with the following data:

Nom	Marque	Mise en circulation	Description	Options	Image
C4	Citroën - France	Dec 11, 2014 12:00:00 AM	Moteur diesel, ABS,	Phares directionnels, radars de recul	

## ➤ Étape 7 :

Voici le code dans Netbeans qui permet d'afficher toutes les voitures créés dans Pimcore. Lorsque l'on crée des classes dans Pimcore, automatiquement un fichier est créé dans Netbeans.



The screenshot shows the NetBeans IDE interface. On the left, the 'Projects' pane displays a file tree for a project named 'voiture'. The tree includes folders for 'config', 'lib', 'models', 'modules', 'static', 'plugins', 'website', 'controllers', 'views', 'areas', 'layouts', 'scripts', 'default', 'voiture', 'var', 'editVoiture.php', 'listeVoiture.php', 'sauvegarderVoiture.php', 'Jhtaccess', and 'index.php'. The 'Source' pane on the right shows the code for 'listeVoiture.php'. The code is a PHP script that uses a foreach loop to iterate over a collection of cars (\$this->voitures) and displays them in an HTML table. The table has columns for Nom, Marque, Mise en circulation, Description, Options, and Image. Each row also includes a nested loop to display the names of the people associated with the car (\$this->voiture->getPersonnes()).

```
1 <h1>Toutes les voitures</h1>
2
3
4 <table border=1>
5
6 <th>Nom</th>
7 <th>Marque</th>
8 <th>Mise en circulation</th>
9 <th>Description</th>
10 <th>Options</th>
11 <th>Image</th>
12 <th>Personnes à bord</th>
13
14 <?php foreach ($this->voitures as $uneVoiture) : ?>
15
16 <tr>
17 <td> <?= $this->escape($uneVoiture->getNom()); ?> </td>
18 <td> <?php echo $this->escape($uneVoiture->getMarque()[0]->getNom()) . ' - ' . $this->escape($uneVoiture->getMarque()[0]->getPays()); ?> </td>
19 <td> <?php echo $this->escape($uneVoiture->getMiseEnCirculation()); ?> </td>
20 <td> <?php echo $uneVoiture->getDescription(); ?> </td>
21 <td> <?php echo $this->escape($uneVoiture->getOptions()); ?> </td>
22 <td> ";
26 } ?>
27 </td>
28 </tr>
29
30 <?php endforeach; ?>
31
32 </table>
```

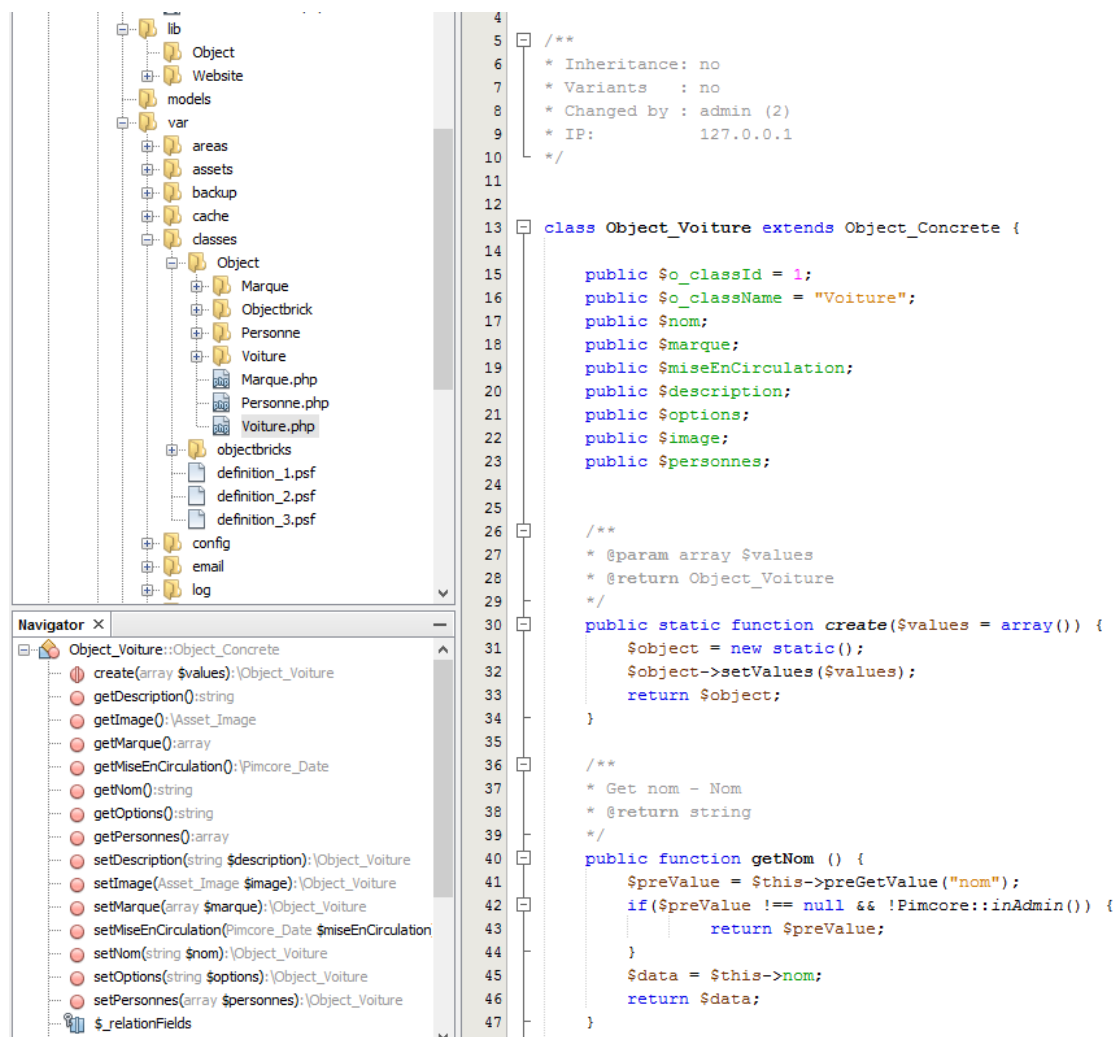


## ➤ Étape 8 :

Dans le premier encadré à gauche, on peut donc remarquer que lorsque l'on a créé nos objets dans Pimcore, les fichiers des trois classes ont été automatiquement générés, il a donc créé tous les setters et les getters ce qui nous permet de les réutiliser sans avoir à les coder, c'est donc un gain de temps. Ce sont donc ces propriétés que l'on réutilise dans la vue précédente afin de récupérer les divers champs.

Concernant la documentation des fonctions codées préalablement par Zend, elle est fournie en même temps ce qui nous permet de savoir quels paramètres elle reçoit ainsi que ce qu'elle doit retourner. Lorsque l'on crée un champ dans Pimcore, par exemple le nom de la voiture, automatiquement NetBeans crée la fonction `getNom()` et ainsi de suite pour tous les champs qui seront créés par la suite.

Sur l'image, on peut observer la partie en bas à gauche comprend toutes les méthodes qui ont été créés, et donc qui correspond aux champs créés dans Pimcore.



The screenshot displays the NetBeans IDE interface. On the left, the 'Project Explorer' shows the project structure, including folders like 'lib', 'models', 'var', and 'classes'. The 'classes' folder is expanded, showing subfolders for 'Object', 'Objectbrick', 'Personne', and 'Voiture'. The 'Voiture' folder contains files like 'Voiture.php', 'Voiturebrick', and 'Voiturebrick.php'. Below the project explorer, the 'Navigator' window shows the 'Object\_Voiture::Object\_Concrete' class, listing various methods such as `create(array $values):Object_Voiture`, `getDescription():string`, `getImage():Asset_Image`, `getMarque():array`, `getMiseEnCirculation():Pimcore_Date`, `getNom():string`, `getOptions():string`, `getPersonnes():array`, `setDescription(string $description):Object_Voiture`, `setImage(Asset_Image $image):Object_Voiture`, `setMarque(array $marque):Object_Voiture`, `setMiseEnCirculation(Pimcore_Date $miseEnCirculation)`, `setNom(string $nom):Object_Voiture`, `setOptions(string $options):Object_Voiture`, `setPersonnes(array $personnes):Object_Voiture`, and `$_relationFields`.

On the right, the 'Code Editor' shows the PHP code for the `Object_Voiture` class, which extends `Object_Concrete`. The code includes a class definition with properties like `$o_classId`, `$o_className`, `$nom`, `$marque`, `$miseEnCirculation`, `$description`, `$options`, `$image`, and `$personnes`. It also includes a `create` static function and a `getNom` function. The `getNom` function is documented with a comment indicating it returns the 'nom' property.

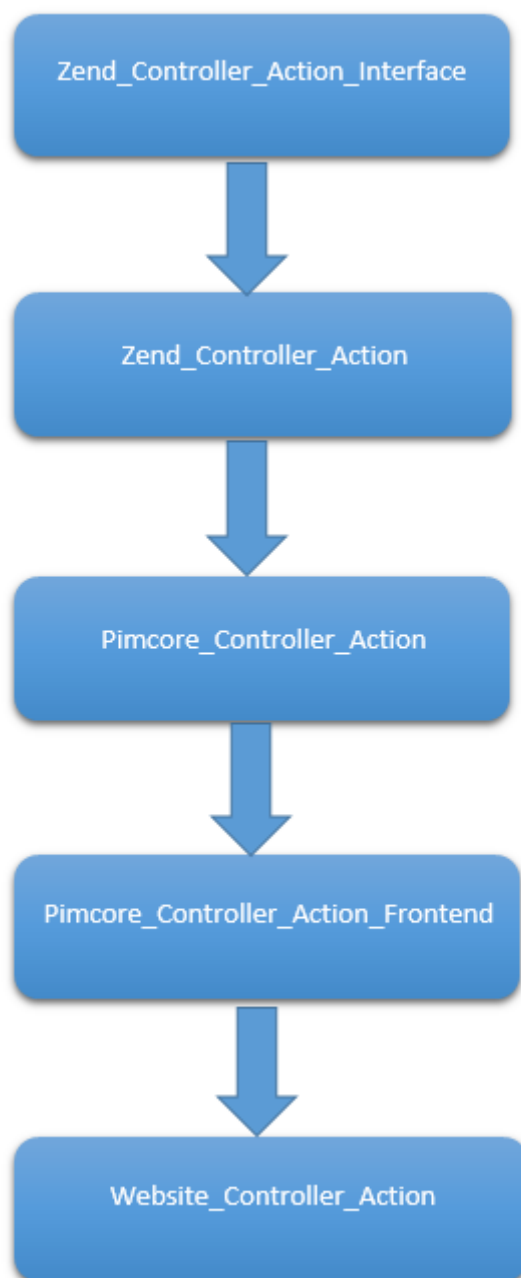
### ➤ Étape 9 :

Voici le contrôleur qui gère la classe Voiture, il contient les différentes actions possibles, ici lister les voitures, les éditer et les sauvegarder. On peut aussi remarquer que la classe VoitureController hérite de la classe Website\_Controller\_Action.

Les deux premières sont identiques, seul les vues changent. La première affichera seulement la liste de voitures alors que la seconde, affichera non seulement la liste des voitures, mais on aura la possibilité de modifier les champs, comme son nom, sa marque etc.

```
<?php
2
3
4 class VoitureController extends Website_Controller_Action {
5
6     public function indexAction() {
7
8         $uneVoiture = new Object_Voiture(); //instanciation
9         $this->view->voitures = $uneVoiture->fetchAll();
10    }
11
12    public function listervoitureAction() {
13
14        $uneVoiture = new Object_Voiture(); //instanciation
15        $items = Object_Voiture::getList(); //recupere la liste des voitures
16        $this->view->voitures = $items;
17    }
18
19    public function editervoitureAction() {
20
21        $uneVoiture = new Object_Voiture(); //instanciation
22        $items = Object_Voiture::getList(); //recupere la liste des voitures
23        $this->view->voitures = $items;
24    }
25
26    public function sauvegardervoitureAction() {
27
28        $uneVoiture = new Object_Voiture();
29        // $uneVoiture->setName("New name");
30        $uneVoiture->save();
31        $this->view->voitures = $items;
32        $redirector = $this->_helper->redirector('index', 'Unauthorized');
33    }
34
35 }
```

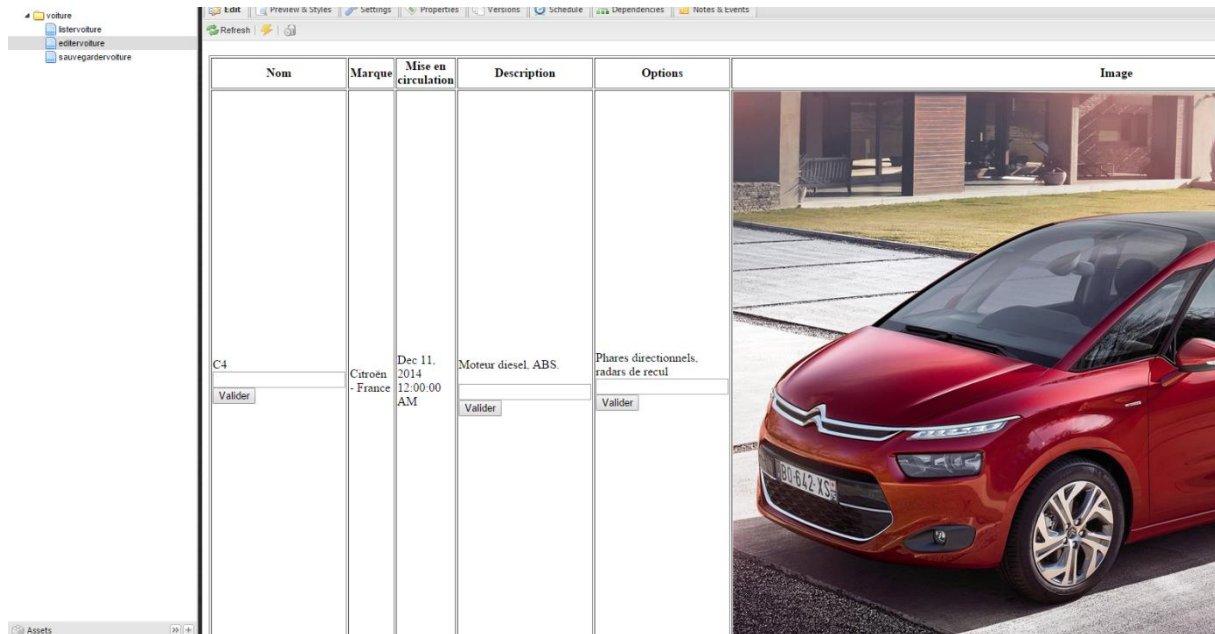
Pour mieux comprendre l'héritage entre les différentes classes, voici un diagramme UML.



Toutes les classes qui héritent de `Website_Controller_Action`, héritent donc de toutes celles qui se situent au-dessus d'elles, cela évite donc le code inutile et permet donc un code PHP « unique ».

## ➤ Étape 10 :

Au lieu de tester dans un navigateur, Pimcore nous permet de visualiser directement ce que rend notre vue que l'on vient de coder dans l'EDI.



Et si jamais il y a une erreur dans le code que nous venons de réaliser, Pimcore nous indique lesquelles et l'endroit où elle se situe.

