|  |
| --- |
| **Module Catalogue** |
| Document Technique |
|  |
| Version 2.0 |
|  |
| **AUGUEY Thomas** |
| **23/02/2013** |
|  |



Contenu

[Introduction 2](#_Toc349379152)

[Termes 3](#_Toc349379153)

[Environnement logiciel 4](#_Toc349379154)

[Arborescence 5](#_Toc349379155)

[Modèle de données 6](#_Toc349379156)

[Entités 6](#_Toc349379157)

[Domaines 7](#_Toc349379158)

[Formats 8](#_Toc349379159)

[Relations 9](#_Toc349379160)

[Acteurs 10](#_Toc349379161)

[Visiteur 10](#_Toc349379162)

[Administrateur 10](#_Toc349379163)

[Système 10](#_Toc349379164)

[Résultats d’opération 11](#_Toc349379165)

[Détail des codes 11](#_Toc349379166)

[Détail des dialogues 11](#_Toc349379167)

[Cas d’utilisation 12](#_Toc349379168)

[Crée un catalogue 13](#_Toc349379169)

[Développement 14](#_Toc349379170)

[Point d’entrée de l’application 14](#_Toc349379171)

[Modèle MVC 14](#_Toc349379172)

[API Références 14](#_Toc349379173)

[Modèle objet et Classes Managers 14](#_Toc349379174)

[Paramètres 15](#_Toc349379175)

# Introduction

Module Utilisateur est utilisé pour implémenter une gestion des comptes et connexions d’utilisateurs pour votre site web.

Fonctionnalités :

* Gestion des comptes utilisateurs
* Gestion des sessions par jeton de connexion de et adresse IP
* Gestion des informations d’indentification (identité)
* Gestion des informations de location (adresses)

# Termes

Webframework

Projet principal de ce module.

# Environnement logiciel

Webframework

Ce module est développé pour le projet **Webframework**.

Votre application web doit au minimum implémenter l’interface IApplication et définir la variable globale $app pour fonctionner normalement (voir le modèle **/wfw/minimal** du projet **Webframework**).

PHP

Ce module est programmé avec **PHP version 5.4.3**.

Les modules suivants doivent être disponibles :

* **php\_pgsql**

Base de données

Le **système de base de données** utilisé est de type **PostgreSQL 8.3**.

Environnement de développement intégré (IDE)

Le module est développé sous l’environnement **NetBeans 7.2.1** avec les langages **PHP/Javascript/XHTML et SQL.**

Suivit de version

L’Utilitaire **GIT** est utilisé pour le suivit de version. La branche master est utilisée pour contenir la dernière version stable du projet.

**Utile** : Le script archive.sh permet de créer une archive compressée du projet ne contenant que les fichiers importants. L’archive est stockée dans le dossier « **../archives »** et nommée automatiquement avec la version du projet en cours.

Modèle de données

La conception de la **base de données** et du **modèle orienté objet** à été réalisé en **UML** avec le logiciel **PowerAMC 15.1**.

**Important** :

Pour générer correctement le code **PHP depuis PowerAMC**, veuillez copier le fichier « documents/php.xol » dans le répertoire d’installation de votre programme **PowerAMC** (ex : C:\Program Files (x86)\Sybase\PowerAMC 15\Fichiers de ressources\Langages objet\php.xol).

Ce fichier permet de générer le modèle orienté objet et les classes **managers** permettant l’interaction avec la base de données.

# Arborescence

Voici une vue d’ensemble des dossiers du projet.

* DOCUMENTS : Documents divers relatif à ce projet
* MCD : Fichiers du Modèle conceptuel de données (PowerAMC)
* nbproject : Fichiers du projet **NetBeans**
* SQL : Fichiers SQL de création de la base de données
* wfw-XXX : Librairies et sources du projet à lier avec votre application

# Modèle de données

Le modèle des données persistantes est conçu sur le modèle Merise.

Le document « **mcd/user.mcd** » propose une version schématisé du modèle conceptuel de données, ce modèle est utilisé pour générer le script d’installation SQL. [Vous pouvez ouvrir ce fichier avec le logiciel **PowerAMC** de **Sybase**].

## Entités

Liste les tables ajoutées à la base de données.

Le tableau des entités est définit ainsi :

* Le **type** de donnée est écrit de façon générique (VARCHAR, CHAR, INTEGER, REAL, …)
* La **contrainte** est appliquée sur les champs par la base de données (Domaines, Clés, Check, …)
* Le **format**, définit plus bas, définit la syntaxe pour les chaines de caractères

CATALOG

Définit un catalogue.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribut | Description | Type | Contrainte | Format |
| CATALOG\_ID | Identifiant | INT | PRIMARY KEY | Integer |
| CATALOG\_TITLE | Titre | VARCHAR(80) | NOT NULL | String |
| CATALOG\_DESC | Description courte | VARCHAR(256) | NOT NULL | String |
| CATALOG\_TYPE |  |  |  |  |

CATALOG\_ITEM

Définit un item*.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribut | Description | Type | Contrainte | Format |
| ITEM\_ID | Identifiant | INT | PRIMARY KEY | Integer |
| ITEM\_TITLE | Titre | VARCHAR(80) | NOT NULL | String |
| ITEM\_DESC | Description courte | VARCHAR(256) | NOT NULL | String |

CATALOG\_ITEM\_CATEGORY

Typage d’un item. Permet de définit le type (ex : PRODUIT) et la catégorie (ex : BANANE)*.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribut | Description | Type | Contrainte | Format |
| ITEM\_CATEGORY\_ID | Identifiant | VARCHAR(80) | PRIMARY KEY | Identifier |
| CATEGORY\_DESC | Description courte | VARCHAR(256) | NOT NULL | String |
| CATEGORY\_TYPE | Typage | ITEM\_TYPE | NOT NULL | Identifier |

## Domaines

CATALOG\_ITEM\_TYPE

Types d’items (étendu par l’application).

|  |  |
| --- | --- |
| Valeur : CHAR(10) | Description |
| GENERIC | Générique |

CATALOG\_TYPE

Types de catalogues (étendu par l’application).

|  |  |
| --- | --- |
| Valeur : CHAR(10) | Description |
| GENERIC | Générique |

## Formats

Tous les formats utilisés sont hérités du projet **Webframework**, reportez-vous au document technique du projet pour plus d’informations.

## Relations

1. CATALOG CATALOG\_ITEM
   1. Un **catalogue** peut être attaché à plusieurs **items**
   2. Un **item** est attaché à un catalogue
2. CATALOG\_ITEM CATALOG\_ITEM\_CATEGORY
   1. Un **item** est typé par une **catégorie**
   2. Une **catégorie** peut typer un ou plusieurs **items**

Operations système

Les opérations systèmes sont conçues sur le modèle UML.

Le document « **mcd/operations.moo** » propose une version schématisé des cas d’opération. [Vous pouvez ouvrir ce fichier avec le logiciel **PowerAMC** de **Sybase**].

## Acteurs

### Visiteur

Utilisateur non enregistré.

### Administrateur

Super-Utilisateur de l’interface de gestion.

## Système

Définition du système avec ses différentes couches de dépendances.

**Système Langage**

1. Application HTML/Javascript
2. Interface Serveur PHP
   1. Système embarqué (système de fichier) Console UNIX
3. Interface de Base de données PG/SQL PL/pgSQL
   1. Base de données SQL SQL

## Résultats d’opération

Les résultats d’opération sont basés sur le modèle du projet **WebframeWork**, pour plus de détails à ce sujet reportez-vous au document technique concerné.

Notez que ces codes d’erreurs viennent s’ajouter aux codes existants du projet **Webframework**. Certains d’entre eux peuvent être réutilisés dans ce projet.

### Détail des codes

**Contexte Code Description**

ERR\_OK CATALOG\_CREATED Catalogue créé

ERR\_OK CATALOG \_DELETED Catalogue supprimé

ERR\_OK ITEM\_CREATED Item créé

ERR\_OK ITEM\_ DELETED Item supprimé

### Détail des dialogues

|  |  |  |
| --- | --- | --- |
| **Contexte** | **Code** | **Description** |
| xxxx | xxx | xxxxxx |

## Cas d’utilisation

Les cas d’utilisations (ou use cases) liste les procédures et leurs interactions avec le système.

Une procédure est soit validée, soit annulée dans sa totalité. Dans le cas contraire et pour maintenir un système cohérant, aucunes des actions précédentes n’est conservées.

Etat de l’implémentation

Cette liste informe sur l’avancement du développement. Un statut OK indique que le cas d’utilisation est rédigé et codé dans l’application, au contraire un statut KO indique un développement en cours

|  |  |  |  |
| --- | --- | --- | --- |
| **Cas d’utilisation** | **Statut** | **Rôle** | **Priorité** |
| catalog\_create | KO | Administrateur | Requis |
| catalog\_create\_item | KO | Administrateur | Requis |
| catalog\_register\_item | KO | Visiteur | Requis |

### Crée un catalogue

Crée un nouveau catalogue.

Informations

Identifiant : catalog\_create

Acteur : Crée un catalogue

Entrée

Type Description

* String Titre
* String Description

Sortie

Type Description

* Name GUID

Procédure

1. Insert une entrée dans la table CATALOG

### 

# Développement

## Point d’entrée de l’application

Conformément au projet **Webframework**, ce projet implémente l’interface IModule.

Vous trouverez le code de la classe CatalogModule dans le dossier « wfw-1.x/lib/catalog » avec l’ensemble des classes du modèle, vous devez inclure le fichier « Catalog.php » à votre application pour pouvoir utiliser les fonctionnalités du module **Catalogue**.

## Modèle MVC

Le projet est construit sur le modèle <**Modèle-Vue-Contrôleur**>.

* Les classes C++ du modèles sont stockées dans le dossier « wfw-1.x/lib **»**.
* Les classes de vues sont stockées dans le dossier « wfw-1.x/view**/catalog**»
* Les classes contrôleurs sont stockées à la racine du dossier « wfw-1.x/**ctrl/catalog ».**

## API Références

La documentation du code source est disponible dans le dossier « doc », il contient une arborescence web (HTML) généré à partir du code source.

Les fichiers de projet **Doxygen** est situé dans le dossier «  doxygen ».

## Modèle objet et Classes Managers

Le modèle de données a son équivalent en classes objets sous **PHP**.

## Paramètres

Voici les différentes constantes de configuration influant sur le module :

|  |  |  |  |
| --- | --- | --- | --- |
| Section | Nom | Plateforme | Usage |
| path | **catalog\_module** | **-** | globals.php |
| Chemin d’accès à la librairie CATALOG | | | |