|  |
| --- |
| **Module IO** |
| Document Technique |
|  |
| Version 1.0 |
|  |
| **AUGUEY Thomas** |
| **05/04/2013** |
|  |



Contenu

[Introduction 2](#_Toc348617853)

[Termes 3](#_Toc348617854)

[Environnement logiciel 4](#_Toc348617855)

[Arborescence 6](#_Toc348617856)

[Modèle de données 7](#_Toc348617857)

[Entités 7](#_Toc348617858)

[Formats 7](#_Toc348617859)

[Operations système 8](#_Toc348617860)

[Acteurs 8](#_Toc348617861)

[Administrateur 8](#_Toc348617862)

[Système 8](#_Toc348617863)

[Résultats d’opération 9](#_Toc348617864)

[Détail des codes 9](#_Toc348617865)

[Détail des dialogues 9](#_Toc348617866)

[Cas d’utilisation 10](#_Toc348617867)

[Envoyer un message 11](#_Toc348617868)

[Développement 12](#_Toc348617869)

[Point d’entrée de l’application 12](#_Toc348617870)

[Modèle MVC 12](#_Toc348617871)

[API Références 12](#_Toc348617872)

[Modèle objet et Classes Managers 12](#_Toc348617873)

[Paramètres de configuration 13](#_Toc348617874)

[Notes d’implémentations 14](#_Toc348617875)

[Dépannage 15](#_Toc348617876)

[Problème de connexion au server SMTP 15](#_Toc348617877)

# Introduction

**Module IO** est utilisé pour la gestion des échanges de données (upload/download).

Fonctionnalités :

* Envoi (Upload) / Téléchargement (Download) de données par fragment

# 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\_fileinfo**
* **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/io.mcd** » propose une version schématisé du modèle conceptuel de données, ce modèle est utilisé pour générer le script d’installation SQL et model PHP/Objet. [Vous pouvez ouvrir ce fichier avec le logiciel **PowerAMC** de **Sybase**].

## Dictionnaire

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nom** | **Code** | **Type de données** | **Longueur** | **Précision** |
| base64\_data | BASE64\_DATA | Texte |  |  |
| begin\_date | BEGIN\_DATE | Date & Heure |  |  |
| checksum | CHECKSUM | Caractère variable (256) | 256 |  |
| client\_ip | CLIENT\_IP | Caractère variable (200) | 200 |  |
| content\_type | CONTENT\_TYPE | Caractère variable (260) | 260 |  |
| file\_size | FILE\_SIZE | Entier |  |  |
| filename | FILENAME | Caractère variable (260) | 260 |  |
| io\_packet\_id | IO\_PACKET\_ID | Entier |  |  |
| io\_packet\_size | IO\_PACKET\_SIZE | Entier |  |  |
| io\_upload\_id | IO\_UPLOAD\_ID | Caractère variable (8) | 8 |  |
| output\_path | OUTPUT\_PATH | Caractère variable (260) | 260 |  |
| packet\_count | PACKET\_COUNT | Entier |  |  |
| packet\_num | PACKET\_NUM | Entier |  |  |
| packet\_size | PACKET\_SIZE | Entier |  |  |
| packet\_status | PACKET\_STATUS | Booléen |  |  |
| upload\_complete | UPLOAD\_COMPLETE | Booléen |  |  |
| upload\_path | UPLOAD\_PATH | Caractère variable (260) | 260 |  |
| upload\_token | UPLOAD\_TOKEN | Caractère variable (8) | 8 |  |

## Liens d'association

|  |  |  |  |
| --- | --- | --- | --- |
| **Association** | **Entité** | **Identifiant** | **Cardinalité du rôle** |
| IO\_stocker | IO\_packet |  | 1,1 |
| IO\_stocker | IO\_Upload |  | 0,n |

## Formats

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

# 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

### 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 IO\_FILE\_UPLOADED Fichier reçu

ERR\_FAILED IO\_FILE\_TO\_BIG Le fichier excède la taille maximale autorisée

ERR\_FAILED IO\_ZERO\_FILE\_SIZE Le fichier possède une taille nulle

ERR\_FAILED IO\_REPOSITORY\_ALREADY\_EXISTS Le dépôt existe déjà

ERR\_SYSTEM IO\_CANT\_CREATE\_UPLOAD\_DIR Le dossier d’envoie n’existe pas et ne peut pas être créé

ERR\_SYSTEM IO\_DUMMY\_UPLOAD\_FILE\_CREATE Le fichier de réception ne peut pas être créé

ERR\_SYSTEM IO\_UPLOAD\_DIR\_NOT\_EXISTS Le dossier d’envoie n’existe pas

ERR\_SYSTEM IO\_CANT\_LINK\_EVENT Impossible de lié un événement à ce dépôt

### Détail des messages

|  |  |  |
| --- | --- | --- |
| **Contexte** | **Code** | **Description** |
| ERR\_SYSTEM | IO\_CANT\_LINK\_EVENT | Veuillez vérifier les autorisations d’accès sur le serveur. |
| ERR\_FAILED | IO\_REPOSITORY\_ALREADY\_EXISTS | Veuillez utiliser un autre identifiant de dépôt |

## 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** |
| mail\_send\_message | KO | Administrateur | Requis |
| mail\_create\_contact | KO | Administrateur | Optionnel |
| mail\_remove\_contact | KO | Administrateur | Optionnel |

### Débute l’upload de données

Prépare le serveur à recevoir des données.

Informations

Identifiant : begin\_upload

Acteur : Visiteur

Entrée

Identifiant Description

file\_size Taille du fichier (en Bytes)

filename Nom du fichier

content\_type Type MIME

mode Optionnel, Type d’upload. Si file est spécifié, le fichier sera écrit sur le disque du serveur en même temps qu’en base de données.

Sortie

Type Description

io\_upload\_id Identifiant de l’upload

packet\_count Nombre de paquet à télécharger

packet\_size Taille de chaque paquet (en Bytes)

Procédure

1. Vérifie la taille minimum/maximum allouée à l'upload (IO\_FILE\_TO\_BIG, IO\_ZERO\_FILE\_SIZE). La taille limite est définit dans la configuration (io\_module::max\_upload\_size)
2. Si mode == file
   1. Vérifie que le répertoire d’upload existe (io\_module::upload\_dir)
3. Initialise l'entrée en BDD
4. Si mode == file
   1. Prépare le fichier de stockage

Note

Pour procéder à l’upload des données. Le programme doit exécuter le contrôleur check\_upload.

### Vérifie l’état d’un upload

Informe sur l’état d’avancement de l’upload et le prochain paquet manquant.

Informations

Identifiant : check\_upload

Acteur : Visiteur

Entrée

Identifiant Description

io\_upload\_id Identifiant de l‘upload

Sortie

Type Description

packet\_offset Offset du paquet (en Bytes)

packet\_num Numéro du paquet

packet\_size Taille du paquet (en Bytes)

Procédure

1. Test l’existence des paquets en base (io\_packet). Si un paquet est manquant, les paramètres de retour sont complétés et la fonction retourne IO\_FILE\_UNCOMPLETED

Note

Cette fonction est à la fois utilisé pour vérifier l’intégralité d’un upload et également pour identifier la prochaine partie manquante du chier.

### Initier un envoi de données

Initialise un envoi de données.

Informations

Identifiant : create\_upload

Acteur : Visiteur

Entrée

Identifiant Description

Mail Adresse du nouvel utilisateur

Identifier Nom d’utilisateur

Password Mot de passe

Name Identifiant client (module Client)

Sortie

Identifiant Description

Procédure

1. Vérifie si l’utilisateur est connecté
2. Récupère le compte utilisateur associé à la connexion (USER\_ACCOUNT**)**
3. Vérifie si un profil existe déjà en base de données pour ce compte utilisateur (APP\_PROFIL\_EXISTS)
4. Initialise les entrées dans les tables IO\_UPLOAD\_PROCESS

### Créer un dépôt de données

Crée un dépôt à partir de données utilisateurs.

Informations

Identifiant : repository\_create

Acteur : Visiteur

Entrée

Identifiant Description

repository\_id Optionel, Identifiant du dépôt

repository\_pwd Optionel, Mot-de-passe associé au dépôt

repository\_type Optionel, Type de dépôt

is\_readonly Optionel, Si true, les données ne sont plus modifiables une fois créées

is\_event Optionel, Créer un lien d’événement sur ce dépôt

use\_data Optionel, Si true, crée un répertoire pour le stockage de fichiers de données

note Optionel, Inutilisé pour le moment

… Optionel, Données du dépôt

Sortie

Identifiant Description

repository\_id Identifiant du dépôt

Procédure

1. Génère le nom de dépôt
2. vérifie si le dossier existe (io\_repository\_already\_exists**)**
3. Crée le document XML avec l'ensemble des données reçues en paramètres (app\_cant\_create\_resource)
4. Crée le dossier de données (app\_cant\_create\_resource)
5. Envoie d’un mail de notification (mail\_module::send\_message)
6. Attache un événement (io\_cant\_link\_event)

### Supprimer une autorisation d’envoie

Initialise un envoi de données.

Informations

Identifiant : delete\_upload

Acteur : Administrateur

Entrée

Type Description

* Mail Adresse du nouvel utilisateur
* Identifier Nom d’utilisateur
* Password Mot de passe
* Name Identifiant client (module Client)

Sortie

Type Description

Procédure

1. Vérifie si l’utilisateur est connecté
2. Récupère le compte utilisateur associé à la connexion (USER\_ACCOUNT**)**
3. Vérifie si un profil existe déjà en base de données pour ce compte utilisateur (APP\_PROFIL\_EXISTS)
4. Initialise les entrées dans les tables IO\_UPLOAD\_PROCESS

# Développement

Ce module suit les recommandations de développement définit par le projet **Webframework**. Pour plus d’informations, consultez la section « **Modules** » disponible dans la documentation technique du projet **Webframework**.

## Paramètres de configuration

Voici les différentes constantes de configuration influant sur le module :

|  |  |  |  |
| --- | --- | --- | --- |
| [io\_module] | Requis | Définition | Exemple |
| ctrl\_path | **OUI** | Chemin d’accès aux contrôleurs | ${io\_module\_path}/ctrl/io |
| upload\_dir | **OUI** | Dossier de stockage temporaire des données (si mode=="file" seulement) | bin/public |
| public\_output\_dir | **OUI** | Dossier de stockage des fichiers publiques (envoies initiés avec le UseCase: begin\_upload) | bin/upload |
| max\_upload\_size | **OUI** | Taille maximal accepté pour les envois publiques (initiés avec le UseCase: begin\_upload)  [Spécifié en bytes] | 52428800 ; 50Mo |
| packet\_size | **OUI** | Taille des paquets accepté pour les envois publiques (initiés avec le UseCase: begin\_upload).  [Spécifié en bytes] | 524288 ; 512Ko |
| storage\_mode | **OUI** | Mode de stockage des données.  Si la valeur file est spécifiée le fichier est construit parallèlement dans le dossier local (upload\_dir) et la **BDD**. | file |

## Procédure de stockage

Pour des raisons de sécurités et de performances, le stockage des fichiers provenant d’ordinateurs clients doivent faire l’objet d’une attention particulière. Le module IO est écrit pour vérifier divers points.

Notamment :

* Eviter la saturation de l’espace disque en refusant des fichiers trop volumineux.
* Contrôler la provenance de la requête (vérification de l’IP cliente et du code anti-spam) afin d’éviter l’abus d’utilisation de bande passante.

**Il existe 2 méthodes pour stocker un fichier sur le serveur :**

1. Avec un accès publique via le contrôler begin\_upload :

Dans ce cas, l’autorisation de stockage est créée automatiquement par l’utilisateur final. Une entrée est insérée dans la table IO\_UPLOAD avec les restrictions suivantes :

* Le dossier de destination est prédéfinit par la variable de configuration [io\_module:public\_output\_dir]
* La taille des données est limitée par la variable de configuration [io\_module:max\_upload\_size]
* La taille des paquets est définit par la variable de configuration [io\_module:packet\_size]

L’utilisateur final peut ensuite utiliser les contrôleurs de flux packet\_upload et finalize\_upload pour initialiser les données.

1. Avec un accès privé via le contrôler create\_upload :

Dans ce cas, l’autorisation de stockage est créée par l’administrateur. Une entrée est insérée dans la table IO\_UPLOAD avec des paramètres de configurations spécifiques. L’utilisateur final peut ensuite utiliser les contrôleurs de flux packet\_upload et finalize\_upload pour initialiser les données.