

AUGURIA



Module Livre / Editeur de livres

## **Références du document :**

<b><i>Rédaction :</i></b>	Patrick Raguin – AUGURIA
<b><i>Type de document :</i></b>	Documentation
<b><i>Date :</i></b>	26/08/2008
<b><i>Nom du document :</i></b>	Documentation_ - _Book_revise.odt

## **Objet du document :**

Documentation sur le module « Book » de gestion de livres et de leurs droits d'auteur.

## **Historique du document :**

<b><i>Version</i></b>	<b><i>Date de modification</i></b>	<b><i>Auteur</i></b>	<b><i>Modification</i></b>
V1.0	26/08/2008	PatrickRaguin	Initialisation du document
V1.1	08/06/10	Pierre Morin	Révision du documen

## **Contact :**

AUGURIA  
Patrick Raguin  
[patrick.raguin@auguria.net](mailto:patrick.raguin@auguria.net)

AUGURIA  
Pierre Morin  
[pierre.morin@auguria.net](mailto:pierre.morin@auguria.net)

AUGURIA  
Cyrille de Lambert  
[cyrille.delambert@auguria.net](mailto:cyrille.delambert@auguria.net)  
Tél : 02.51.13.50.12

# **I. Introduction**

L'objectif du module Livre est de répondre aux besoins des éditeurs de livres.

Cela intègre :

- La gestion automatisée du droit de prêt
- Le stock et la gestion des commandes de livres
- La gestion des droits d'auteurs versés aux éditeurs de création

# Table des matières

I.Introduction.....	3
Fonctionnalités.....	5
1. Activation / désactivation du module Book.....	5
2. Configuration du module.....	5
3. Gestion des livres.....	6
Documentation technique.....	8
1. Base de données.....	8
2. Code PHP.....	8
1) Book.....	8
a) DAO (Data Access Object) .....	8
b) Service.....	9
c) Librairies.....	11
d) Ecrans.....	13
2) Contract.....	15
a) DAO.....	15
b) Service.....	15
c) Ecrans.....	15
3) ContractBill.....	16
a) DAO.....	16
b) Service.....	16
c) Ecrans.....	16

## .Fonctionnalités


### 1. Activation / désactivation du module Book

L'option s'active et se désactive dans la configuration des modules. Pour fonctionner le module Book a besoin des modules suivants :


- Produit
- Facture
- Tiers
- Commercial
- Commandes clients
- Stock
- Composition (de produits)

### 2. Configuration du module

Pour fonctionner correctement, le module a besoin d'être configuré.

Pour cela, après l'activation du module, cliquer sur le bouton de configuration : 

Remplir les champs avec les données propres à votre société :

 Configuration des propriétés du module book Retour liste des modules

Les informations suivantes seront affichées en bas des courriers annuels de droits d'auteur.

Paramètres	Valeur	
Slogan de la société	<input type="text" value="My society slogan"/>	<a href="#">Modifier</a>
Informations légales sur la société	<input type="text" value="TEL. 01 02 03 04 05, FAX 05 06 07 08 09 - SARL AU CAPITAL DE 7"/>	<a href="#">Modifier</a>
Autres informations sur la société (site Internet, par exemple)	<input type="text" value="www.mysociety.com"/>	<a href="#">Modifier</a>

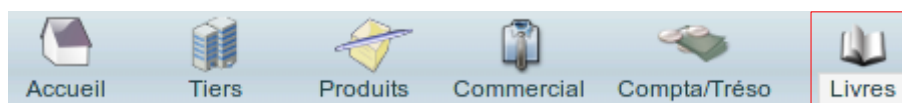
Attention, il est impératif de configurer le module de gestion des stocks pour qu'il « Incrémente les stocks physiques sur ventilation manuelle de la réception des commandes fournisseurs dans les entrepôts » :

 Configuration module stock
 Retour liste des modules

Paramètres	Valeur
Créer un stock/entrepôt propre à l'utilisateur automatiquement à la création de cet utilisateur	Non <input type="button" value="Modifier"/>
<b>Règle de gestion des décréments de stock</b>	
Décrémente les stocks physiques sur validation des factures/avoirs clients (attention, dans cette version, c'est toujours dans le premier entrepôt que se fait l'ajustement)	Non <input type="button" value="Modifier"/>
Décrémente les stocks physiques sur validation des commandes clients (attention, dans cette version, c'est toujours dans le premier entrepôt que se fait l'ajustement)	Non <input type="button" value="Modifier"/>
<b>Règle de gestion des incréments de stock</b>	
Incrémente les stocks physiques sur validation des factures/avoirs fournisseurs (attention, dans cette version, c'est toujours dans le premier entrepôt que se fait l'ajustement)	Non <input type="button" value="Modifier"/>
Incrémente les stocks physiques sur approbations des commandes fournisseurs (attention, dans cette version, c'est toujours dans le premier entrepôt que se fait l'ajustement)	Non <input type="button" value="Modifier"/>
Incrémente les stocks physiques sur ventilation manuelle de la réception des commandes fournisseurs dans les entrepôts	Oui <input type="button" value="Modifier"/>

### 3. Gestion des livres

Après l'activation du module, un nouvel élément apparaît dans le menu du haut de page :



Le menu de gauche est alors le suivant :



Ce nouveau menu va permettre de faire la gestion des livres ainsi que des contrats.

L'entrée de menu « Nouveau livre » permet d'ajouter un livre en remplissant les informations classiques d'un produit ainsi que d'autres informations spécifiques aux livres tels que :

- ISBN (les autres codes ISBN13 et EAN13 sont calculés à partir de l'ISBN saisie)  
L'ISBN doit être sous la forme 2-86889-006-7. Le dernier chiffre correspond à la clé qui peut ne pas être saisie et sera donc calculé en conséquence
- Nombre de pages
- Format

ISBN	<input type="text"/>	ISBN-13	Calculated
EAN	Calculated	EAN13	Calculated
Nombre de pages	<input type="text"/>	Format	Format 2A0 <input type="button" value="v"/>

Dans la fiche d'un produit, un nouvel onglet apparaît : «Livres». Cet onglet permet d'accéder à l'écran de gestion du livre.

Fiche produit		Fiche	Prix clients	Prix fournisseurs	Photos	Statistiques	Références	Fichiers joints	Stock	Livre	Composition de produits
Ref.	Endymion										
Libellé	Endymion										
ISBN	2 221 08856 1										
ISBN13	978-2-221-08856-7				EAN13	8782221088569					
État	<input checked="" type="checkbox"/> En vente				Poids	357 g					
Nombre de pages	576				Format	Format 2A0					
Prix de vente HT	9,45 HT				Prix de vente TTC	9,96975 TTC					
Prix de revient HT	2,5165 HT (18,50715791 Fr)				Prix de revient TTC	2,83767 TTC (18,61389500 Fr)					
Coût de revient total HT	2,5165 HT (18,50715791 Fr)				Coût de revient total TTC	2,83767 TTC (18,61389500 Fr)					
Taux TVA	5,5 %										
Stock réel	0				Seuil d'alerte	500					
Quantité vendue											
Description	Endymion est un roman de science-fiction, genre space opera, écrit par Dan Simmons en 1995 et publié en France en 1996. Ce roman est le premier volet d'un récit qui se terminera avec L'Éveil d'Endymion. Ce roman est aussi le troisième volume des Cantos d'Hypérion, un cycle composé de quatre romans : Hypérion, Le Chute d'Hypérion (The Fall of Hyperion, 1890), Endymion (1895) et L'Éveil d'Endymion (The Rise Of Endymion, 1897), complétés par deux nouvelles : Les orphelins de l'hélice et La mort du centaure. Ce cycle est considéré par certains comme une des œuvres majeures de la science-fiction.										
Note	A lire.										
<a href="#">Editer</a>											
Liste des compositions											
Libellé	Prix unitaire HT	Quantité	Gestion des stocks	Prix de revient	Prix total HT	Prix total TTC					
Couverture pour livre a3	1,00 HT	1	Oui	Oui	1,00 HT	1,196 TTC					
Feuille format A6 de 1g	0,001 HT	288	Oui	Oui	0,288 HT	0,3456 TTC					
cda_Endymion_2010_2	1,2285 HT	1	Oui	Oui	1,2285 HT	1,29607 TTC					

#### Contrat

Ref.	cda_Endymion_2010_2		Editeur	Flymaria	
Début du contrat	01/07/2010		Fin du contrat	31/12/2010	
Taux	13%		Quantité	700	
Prix HT	1,2285		Prix TTC	1,29607	
<a href="#">Courrier annuel</a> <a href="#">Nouveau contrat</a>					
Courriers annuels					
Référence	Début période	Fin période	Qté commandée	Qté vendue	
cda_Endymion_2010_2010	01/01/2010	31/12/2010	0	0	
Anciens contrats					
Taux	Début du contrat	Fin du contrat	Editeur		
14.30	01/07/2010	31/12/2010	Flymaria		

# **.Documentation technique**

## **1. Base de données**

La base de données voit apparaître trois nouvelles tables :

« llx\_product\_book », « llx\_product\_book\_contract » et  
« llx\_product\_book\_contract\_bill ».

- llx\_product\_book : contient les livres
- llx\_product\_book\_contract : contient les contrats
- llx\_product\_book\_contract\_bill : contient les courriers annuels aux auteurs

## **2. Code PHP**

Le dossier contenant le module se trouve à cet emplacement : htdocs/book

Le module Book a en partie été généré à partir du générateur de module Dolibarr UML2Dolibarr (voir détails sur le forum Dolibarr). On retrouve donc la même organisation de code que celui du générateur à savoir, le dao (Data Access Object, Classe d'accès aux données), le service, les écrans et les templates.

Le texte suivant reprend en partie des extraits de la documentation du générateur de modules Dolibarr.

On peut cependant découper le module en 3 parties, chacune organisée de la même façon.

### **1) Book**

#### **a) DAO (Data Access Object)**

Fichier : htdocs/book/class/data/dao\_book.class.php

Les DAO (Data Access Object) permettent d'accéder aux objets stockés en base (ici les livres) en lecture, création, modification, et suppression.

Il faut commencer par instancier le DAO ( `$book_dao = new dao_book($db);` ). On passera en paramètre l'objet d'accès à la base de données. Ensuite, pour charger en mémoire un objet stocké en base, il faudra utiliser la méthode `$book_dao->fetch($id)` où `$id` est le rowid du livre à charger.

L'occurrence souhaitée de l'objet étant stockée en mémoire, on peut alors utiliser les accesseurs (`get<nomAttribut>()` et `set<nomAttribut>($valeur)`) afin de lire et modifier chaque attribut.

Pour qu'une modification persiste en base, on utilisera la méthode `$book_dao->update()`. Elle prend un attribut optionnel, qui indique si l'action doit passer outre les triggers mis en place (1) ou laisser leur exécution normale si nécessaire (0, valeur par défaut).



Pour créer un nouvel objet, le DAO doit être instancié, puis les attributs renseignés (méthode « `$book_dao->set<nomAttribut>($valeur)` »). La méthode `$book_dao->create()` pour alors être utilisée pour générer l'entrée en base de la nouvelle composition.

Enfin, la méthode `$book_dao->delete()` supprime l'objet de la base de données. Un identifiant peut être passé en paramètre pour supprimer directement l'objet ayant le rowid indiqué, sinon, la valeur de l'attribut rowid de l'objet actuellement chargé en mémoire est utilisée.

Autres méthodes :

- `initAsSpecimen()` : initialise l'objet avec des valeurs vides. Rowid à 0.
- `Select ($DB,$fields=",$where=",$order=")`
  - Description : réalise un et retourne le résultat de la requête.
  - Paramètres
    - `$DB` objet permettant de manipuler la base de données
    - `$where` contenu de la clause WHERE sans le mot clé WHERE
    - `$order` contenu de la clause ORDER BY sans le mot clé ORDER BY
  - retour : résultat du `$db->query(requête)` où requête est la requête générée par la fonction.
- `selectQuery($DB, $query,$where=",$order=")` : Idem que `Select`, la requête personnalisée doit être passée en paramètre dans `$query`.
- `exist($id)` : Vérifie que l'id du produit passé en paramètre est bien un livre.

## **b) Service**

Fichier : `htdocs/book/class/business/service_book.class.php`

Le but du service est de fournir un maximum de fonctionnalités pour manipuler l'entité (composition). Il s'appuie largement sur le DAO pour la persistance des données.

Pour être utilisé, un service doit être instancié. Il prend comme paramètre obligatoire la variable de la base de données. On peut ensuite utiliser les méthodes suivantes :

- `getAllListDisplayed($where=",$order=")`
  - Description : Cette méthode est utilisée dans les écrans d'affichage de liste. Elle retourne pour l'entité un tableau contenant toutes les colonnes devant être affichées dans une liste. Pour générer la requête, la méthode fait appel à la fonction statique du dao (`Select`).
  - Paramètres :
    - `$where` : chaîne de caractères avec le contenu de la clause WHERE sans le mot clé WHERE
    - `$order` : chaîne de caractères avec le contenu de la clause ORDER BY sans le mot clé ORDER BY
- `getListDisplayedFields()`
  - Description : Cette méthode est utilisée pour générer les en-têtes du tableau d'affichage. Elle permet de connaître quels sont les champs à afficher parmi ceux sélectionnés.
  - Retour : tableau des `nom_champ => tableau (`
    - `"attribute" => // nom du champ`
    - `"type" => // type (pour le filtre de recherche)`
    - `"label" => // traduction à afficher``)`

- add(), update(), delete()
    - Description : Ces méthodes effectuent les traitements nécessaires pour ajouter, modifier, supprimer l'entité à partir des données POST récupérées sur les formulaires (respectivement) d'ajout, de modification, d'affichage de la fiche.
    - Paramètre :
      - \$post : la variable \$\_POST contenant les données à traiter.
    - Retour : l'identifiant de l'objet en cas de succès (0 pour la suppression) ou -1 en cas d'erreur, l'erreur pouvant être récupérée par getError().
  - getAttributesShow(\$id)
    - Description : Retourne les informations pour afficher la fiche de l'occurrence d'un objet.
    - Paramètre :
      - \$id : identifiant de l'occurrence
    - Retour : tableau des traduction\_nom\_du\_champ => valeur\_formatée pour chaque attribut.
  - getAttributesAdd(), getAttributesEdit()
    - Description : Ces méthodes retournent un tableau permettant de générer le formulaire d'ajout/modification de l'entité.
    - Retour : tableau contenant les informations nécessaires pour le formulaire :
- ```

array(
    array(
        "entity" => "nom de l'entité",
        "attribute" => "nom de l'attribut",
        "type" => type de l'attribut,
        "value" => valeur de l'attribut,
        "label" => traduction à afficher
    )
);

```
- confirmDeleteForm()
    - Retourne le code HTML pour afficher le formulaire de confirmation précédant la suppression.
  - isLivres(\$id)
    - Vérifie que l'id du produit passé en paramètre est bien un livre. Fait appel à la fonction exist du dao.
  - GetFactoryPrice(\$product\_id)
    - Description : Cette méthode calcule le coût de revient avec les compositions dont l'option « coût de revient » est à 1.
    - Paramètre :
      - \$product\_id : rowid du produit (produit fini)
    - Retour : tableau contenant le montant HT et TTC
  - getQtySold(\$product\_id,\$date\_deb=",\$date\_fin=")
    - Description : Calcule le nombre de livres vendus sur la période

### c) *Librairies*

Fichier : htdocs/book/lib/book.lib.php

- getFieldToKeyFieldArray(*\$query*)
  - Retourne un tableau permettant d'associer chaque champ d'une requête au champ correspondant à l'identifiant. Il permet de générer automatiquement les liens vers des entités étrangères.

Exemple:

```
SELECT
  t1.rowid,
  t1.libelle as t1Libelle,
  t1.note as t1Note,
  t2.rowid as t2Rowid,
  t2.libelle as t2Libelle
FROM
  table1 t1,
  table2 t2
```

Retournera le tableau suivant:

```
array(
    [rowid] => rowid,
    [t1Libelle] => rowid,
    [t1Note] => rowid,
    [t2Rowid] => t2Rowid,
    [t2Libelle] => t2Rowid
)
```

- formatValue(*\$type*,*\$value*)
  - retourne la valeur (passé en paramètre) formatée selon le type
- getHtmlForm(*\$type*,*\$attribute\_name*,*\$value*='',*\$null*=0,*\$form\_name*='',*\$textarea\_rows*=3)
  - paramètres
    - *\$type* type of the value
    - *\$attribute\_name* name of the attribute
    - *\$value* value of the attribute
    - *\$null* 0=no null, 1=null
    - *\$form\_name* name of the form
    - *\$textarea\_rows* number of rows to display if the type is 'textarea'
  - Retourne le code HTML correspondant au type passé en paramètre. Le code retourné est utilisable pour un formulaire (ex: input. yes/no, etc.).

Fichier : htdocs/book/lib/barcode.lib.php

- calculate\_isbn\_key(*\$isbn*)
  - paramètre
    - *isbn* International Standard Book Number
  - Retourne la clé d'un numéro International Standard Book Number

- calculate\_ean\_key(\$ean)
  - paramètre
    - ean EAN
  - Retourne la clé d'un numéro EAN 13
- convert\_code\_barre(\$typeIn,\$typeOut,\$value)
  - paramètres
    - typeIn type en entrée
    - typeOut type en sortie
    - value code barre
  - Retourne le code barre sous le format typeOut
- convert\_isbn\_to\_isbn13(\$isbn)
  - paramètre
    - isbn International Standard Book Number
  - Retourne le code ISBN-13 du code ISBN passé en paramètre
- convert\_isbn\_to\_ean13(\$isbn)
  - paramètre
    - isbn International Standard Book Number
  - Retourne le code EAN13 du code ISBN passé en paramètre
- convert\_isbn13\_to\_isbn(\$isbn13)
  - paramètre
    - isbn13 International Standard Book Number 13
  - Retourne le code ISBN du code ISBN-13 passé en paramètre
- convert\_isbn13\_to\_ean13(\$isbn13)
  - paramètre
    - isbn13 International Standard Book Number 13
  - Retourne le code EAN13 du code ISBN-13 passé en paramètre
- convert\_ean13\_to\_isbn13(\$ean13)
  - paramètre
    - ean13 EAN13
  - Retourne le code ISBN-13 du code EAN13 passé en paramètre

**Pas terminée !**

- getGroupISBN(\$isbn)
  - paramètre
    - isbn International Standard Book Number
  - Retourne le Group Identifier du code ISBN
- getSizeGroupISBN(\$isbn)
  - paramètre
    - isbn International Standard Book Number
  - Retourne la taille du code ISBN

« convert\_ean13\_to\_isbn(\$ean13) » n'a pas été créée car elle est interdite.  
 (voir : [http://fr.wikipedia.org/wiki/ISBN#ISBN-13\\_et\\_code\\_.C3.A0\\_barres\\_EAN-13](http://fr.wikipedia.org/wiki/ISBN#ISBN-13_et_code_.C3.A0_barres_EAN-13))

## **d) Ecrans**

### **- *htdocs/book/index.php***

Le fichier index.php contient le code nécessaire à l'affichage de la liste en prenant en compte le filtrage et le tri.

Après initialisation des variables, on instancie un service de l'objet à afficher pour pouvoir le manipuler. Puis on renseigne les colonnes à afficher à l'aide du service (`$service->getListDisplayedFields()`), on initialise la chaîne des paramètres de GET qui servira au filtrage, et on renseigne la traduction du nom de l'entité.

Pour générer la liste des données à afficher, il faut fournir au service de l'entité deux informations : le contenu de la clause WHERE, et le contenu de la clause ORDER BY. Ces 2 clauses sont générées, puis la méthode

`$service->getAttributesShow()` est appelée. Les données récupérées sont ensuite envoyées au template qui pourra les traiter. (voir la partie sur les services pour plus de détails)

Afin de générer la clause ORDER BY, le script utilise les variables GET 'sortfield', 'sortorder' et 'entity'.

- 'sortfield' contient le champs à trier
- 'sortorder' contient 'asc' ou 'desc'
- 'entity' permet de désigner sur quelle entité il faut appliquer le tri à la requête (dans le cas où la page affiche plusieurs listes)

Afin de générer la clause WHERE (pour le filtrage), le script utilise les variables GET qui correspondent aux noms des champs de recherche de la page. Le nom d'un champs de recherche est la concaténation du nom de l'entité et de la colonne. Par exemple 'compteurlibelle'. Pour tous les champs à afficher, on test si `$_GET['entitécolonne']` est renseigné, et si oui on ajoute la condition dans la chaîne \$where. Cela tient compte du type de donnée à comparer : Boolean, Date, autre (comparaison littérale).

### **- *htdocs/book/show\_book.php***

Pour l'affichage des données, cette page se contente d'appeler la fonction du service (`$service->getAttributesShow($_GET['id'])`) puis de retransmettre les informations au template.

Cette page permet aussi selon les droits et les écrans existants de modifier l'objet et de le supprimer. L'édition renvoie vers le ScreenEdit correspondant à l'entité. La suppression se fait via l'envoi de paramètres GET dans la même page. `$_GET["action"] == 'delete'` entraîne l'affichage d'un formulaire de confirmation de la suppression, puis `($_POST["action"] == 'confirm_delete') && ($_POST["confirm"] == 'yes')` entraîne la suppression réelle de l'entité.

Si une intégrité référentielle n'est plus respectée lors de la suppression, un message d'erreur s'affiche. Il convient au développeur de modifier la méthode de suppression de l'objet (`$service->delete($_GET['id'])`) afin de s'assurer que la suppression se fasse en cascade ou soit annulée selon le contexte.

### **- `htdocs/book/add_book.php`**

Cette page affiche le formulaire de modification de l'entité si l'utilisateur en a les droits.

La page fait appel à la fonction « `getAttributesAdd()` » du service dont les données retournées sont stockées dans la variable `$data`. Cette variable est ensuite envoyée au template. La page va ensuite faire appel à la fonction « `getAllFkDisplayed()` » pour retourner les valeurs des listes déroulantes correspondant aux clés étrangères.

Lorsque l'utilisateur clique sur le bouton « Add », la page est rechargée mais cette fois, la variable `$_POST` contient toutes les informations saisies dans le formulaire. La condition « `$_POST[" action "] == " add "` » entraîne l'appel de la fonction « `add` » sur service avec comme paramètre, la variable `$_POST`. Si l'identifiant retourné par l'appel de cette fonction est supérieur à -1, l'affichage est redirigé vers la page show de l'élément inséré.

### **- `htdocs/book/edit_book.php`**

La page de modification est structurée de la même manière que la page d'insertion. En effet, la seule différence se situe au niveau des méthodes appelées. Pour récupérer la liste des éléments à saisir, la page fait appel à la fonction « `getAttributesEdit()` ». Le contrôle de la validation du formulaire se fait grâce à la condition suivante « `$_POST[" action "] == " update "` ». Si cette condition est vérifiée, la page appelle la fonction « `update` » du service. En cas d'erreur ou d'informations manquantes, le formulaire est réaffiché avec le message d'erreur ainsi que les valeurs déjà saisies.

## **2) Contract**

La partie « contract », qui gère les contrats, est organisée de la même manière que la partie book à l'exception qu'il n'existe pas de fichier pour l'affichage et l'édition puisqu'il n'est pas possible de modifier un contrat et que l'affichage se fait dans la partie book.

### **a) DAO**

La classe dao\_book\_contract est similaire à celle du book (dao\_book).

htdocs/book/class/data/dao\_book\_contract.class.php

### **b) Service**

htdocs/book/class/business/service\_book\_contract.class.php

Similaire au service book à l'exception de la méthode :

- newContract(\$post) : Cette méthode crée le contrat ainsi que le produit cad\_<Ref> qui correspond au droit d'auteur. En même temps que le produit sera créé, une commande sera passée et le stock incrémenté du nombre de droits autorisés.

### **c) Ecrans**

Seul l'écran d'ajout est présent (htdocs/book/add\_contract.php).

Il fonctionne de la même façon que la page d'ajout de book. Il affiche également la liste des anciens contrats.

### 3) ContractBill

La partie « contractBill » gère les courriers annuels. est organisée de la même manière que la partie book à l'exception qu'il n'existe pas de fichier pour l'affichage et l'edition puisqu'il n'est pas possible de modifier un contrat et l'affichage se fait dans la partie book.

#### a) DAO

La classe dao\_book\_contract\_bill est similaire à celle du book (dao\_book).

htdocs/book/class/data/dao\_book\_contract\_bill.class.php

#### b) Service

htdocs/book/class/business/service\_book\_contract\_bill.class.php

Similaire au service book à l'exception des méthodes :

- getQteStock(\$id\_contract)
  - Calcul la quantité en stock pour le produit droit d'auteur
- newCommande(\$contract,\$qtyToOrder)
  - Créé une commande fournisseur afin de retrouver un solde nul en stock pour le produit droit d'auteur. Incrémente donc également le stock
- generate(\$id,\$post)
  - Génère le courrier pour un produit. Vérifie qu'il n'a pas été généré et fait ensuite appel à la méthode newCommande.
  - Pour générer le pdf, elle fait appel à la méthode Generate de la class pdf\_courrier\_droit\_editeur (htdocs/includes/modules/book/courrier\_droit\_auteur.class.php)
- generateAll(\$id,\$post)
  - Meme méthode que generate mais cette fois pour tous les produits sous contrats
  - Fait la concaténation de tous les pdf pour la période. Si le pdf n'existe pas il fait appel à la fonction generate pour le produit en question.

#### c) Ecrans

Seul l'écran d'ajout est présent (htdocs/book/add\_book\_contract\_bill.php).