2016-
2017

Dossier de maintenance

PROJET RQS

CARDOEN FLORENT & JEREMY BASTIN

SYS

CreateUsers.sql

Creation des utilisateurs. Crée un role "CBROLE" contenant tous les droits nécessaires. Création du Directory vers "/home/oracle/sgbd"

CreaACLIMG.sql

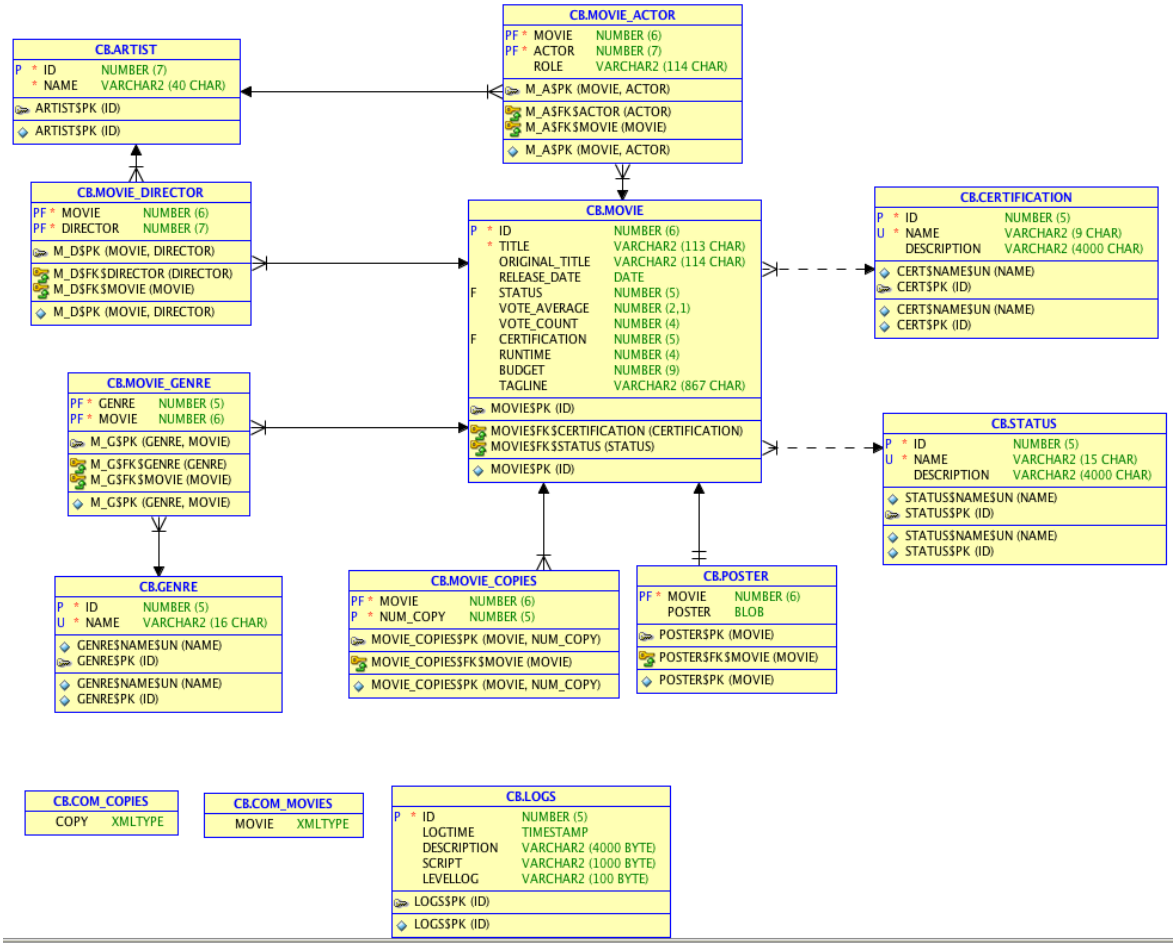
Création de l'autorisation pour l'utilisateur CB à télécharger les posters de films depuis le site image.tmdb.org

CreaAclAllHosts.sql

Création de l'autorisation à l'utilisateur CC à accéder à des adresses extérieures. Ce genre d'autorisation était nécessaire du à notre environnement de développement. En production, CC ne doit avoir accès que à l'adresse du serveur web qui gère la la base de données BP.

CB

Modèle relationnel de données



CreaCB.sql

Création du schéma de la base de données CB. Les tailles des champs a été déterminé avec des calculs statistiques sur la base de données CI.
Création du DBLink vers CC. Ce DBLink est utilisé lors de l'envoi des copies entre CB et CC.

CreaMovieExt.sql

Création de la table Movie_ext (CI). Elle comporte la liste complète des films (300 000).

CreaLog.sql

Création de la table "LOGS", séquence de log (sequence_log) et d'un déclencheur qui augmente le numéro de la séquence pour l'id de la table log.
Création de la procédure de "LOG_INFO".
Paramètre entrant : message du log, endroit du log, niveau de log.
Process : ajoute un tuple dans la table "LOGS"

CreaTrigger.sql

Création d'un trigger qui vérifie que l'année du film soit cohérente.
Si le statut du film est "Released", une erreur est déclenchée si l'année du film est supérieur à l'année en cours augmenté de 5 ans.

AnalyseCIPackage.sql

Ce package est surtout appelé par AlimCB.

FUNCTION GETGENRES

Paramètre entrant : une chaine de caractère contenant la liste de tous les genres d'un film
Sortie : Une table avec les genres récupérés
Process : Découpe la chaine de caractères avec une regex.

FUNCTION GETACTEURS

Paramètre entrant : une chaine de caractère contenant la liste de tous les acteurs d'un film.
Sortie : Une table avec les acteurs récupérés.
Process : Découpe la chaine de caractères avec une regex.

FUNCTION GETDIRECTORS

Paramètre entrant : une chaine de caractère contenant la liste de tous les réalisateurs.
Sortie : Une table avec les réalisateurs récupérés.
Process : Découpe la chaine de caractères avec une regex.

AlimCBPackage.sql

Permet d'insérer des films dans CB à partir de CI

PROCEDURE SANITIZE_FIELDS

Paramètres entrant : un tuple de CI (pointeur)
Process : Récupère les tailles des champs de CB. Nettoie les chaines de caractères, tronque si nécessaire.

PROCEDURE SANITIZE_ACTEURS

Paramètres entrant : liste d'acteurs d'un film (pointeur)
Process : Récupère les tailles des champs de CB. Nettoie les chaines de caractères, tronque si nécessaire.

PROCEDURE SANITIZE_GENRES

Paramètres entrant : liste de genres d'un film (pointeur)
Process : Récupère les tailles des champs de CB. Nettoie les chaines de caractères, tronque si nécessaire.

PROCEDURE SANITIZE_PRODUCTEURS

Paramètres entrant : liste des réalisateurs d'un film (pointeur)
Process : Récupère les tailles des champs de CB. Nettoie les chaines de caractères, tronque si nécessaire.

PROCEDURE ALIMCB

Paramètres entrant : table avec liste des ids
Process : Récupère les tuples depuis CI dont les ids sont dans la table d'ids données en paramètre. Lance ensuite la procédure d'import.

PROCEDURE ALIMCB

Paramètres entrant : nombre de tuples aléatoires

Process : Récupère un nombre aléatoire de tuples depuis CI. Lance ensuite la procédure d'import.

PROCEDURE IMPORTCB

Paramètre entrant : table contenant les tuples à insérer dans CB
Process : Crée tous les tuples correspondant aux films reçus en paramètres et les insère dans CB.

Création d'un nombre aléatoire de copies pour les films.

L'envoie des films vers CC est lancé à la fin de la procédure.

AlimCCPackage.sql

PROCEDURE PREPARE_MOVIE_TO_COPY

Paramètre entrant : id du film à traité
Process : Crée un document XML afin de contenir toutes les informations relatives à un film en y incorporant son statut, sa certification, ses acteurs, ses genres et ses directeurs.
La procédure sélectionne un nombre de copies suivant une distribution uniforme de 0 à N/2. N représentant le nombre total de copies disponibles dans CB. La procédure crée alors un fichier XML pour chaque copie à envoyer. Les documents XML sont ensuite insérés dans la table COM_MOVIES et COM_COPIES dans lesquelles CC pourra venir les récupérer.

PROCEDURE JOB;

Procédure utilisée par les jobs. Celle-ci va appeler la procédure PREPARE_MOVIE_TO_COPY pour chaque film présent dans la table MOVIE.
Ceci fait, la procédure appelle RECEIPT_MOVIES contenu dans CC afin que celui-ci vienne récupérer les films et copies.
La procédure exécute également ARCH_PROG et RETOUR_COPIES contenus dans CC ainsi que RECEIPTION_RETOUR_COPIES. On demande à CC d'archiver ses programmations finies pour qu'on puisse ensuite récupérer les copies qui doivent être retournées à CB.

CreaComTable.sql

Création des tables de communication qui vont de CB à CC. CC viendra récupérer les tuples insérés dans cette table lors d'AlimCC.

ReceiptionRetourCopie.sql

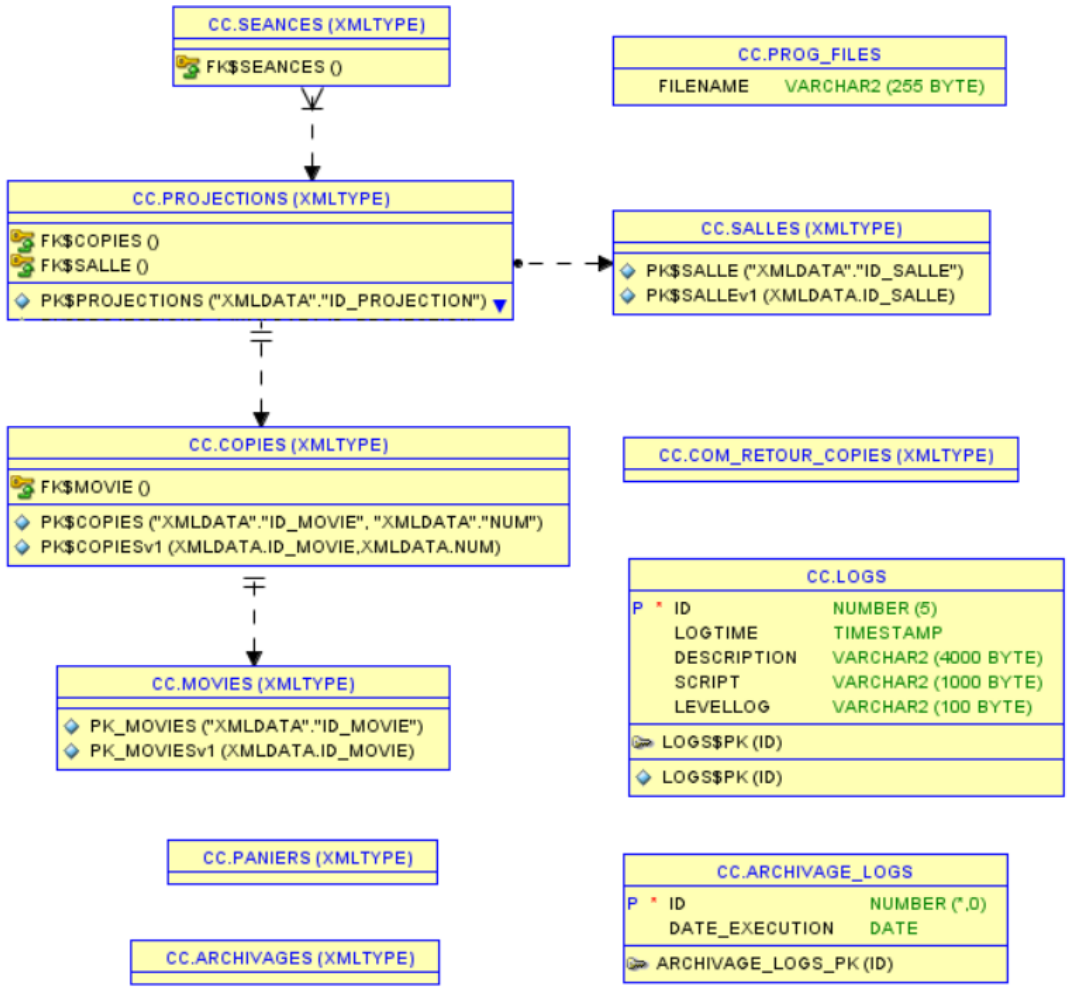
Création d'une procédure dans CB. Celle-ci va récupérer les copies de CC qui doivent être retournées dans CB. Pour cela, elle récupère les copies présentes dans la table de communication COM_RETOUR_COPIES en utilisant le CC_LINK. Etant donné que les copies sont au format XML, la procédure traite les données afin de pouvoir les replacer dans la table MOVIE_COPIES de CB qui est au format relationnelle. Pour finir, la procédure vide la table de communication.

CreaJobAlimCC.sql

Création d'une tache planifiée toutes les semaines qui envoie des films à CC.

CC

Modèle relationnel des données



RegisterXSD.sql

Enregistrement des différents XSD contenus dans le directory (MYDIR)
On enregistre les schémas correspondant à un film, une copie, une programmation, une projection, une archive, un panier, un feedback, une salle et une séance.

CreateXMLTable

Création des différentes tables XML sur bases des schémas XSD précédemment enregistrés.

Les tables crées sont :

Film

<movie>
<id>273159</id>
<title>Donald Takes a Holiday</title>
<original_title>Donald Takes a Holiday</original_title>

<release_date>1986-01-01</release_date>
<status>
<id>401</id>
<name>Released</name>
</status>
<vote_average>0</vote_average>
<vote_count>0</vote_count>
<certification>
<id>561</id>
<name>X</name>
</certification>
<genres/>
<actors/>
<directors/>
</movie>

Copie

<copy>
<movie>126375</movie>
<num>1</num>
</copy>

Projection

<projection>
<idProjection>1282762075</idProjection>
<idMovie>41524</idMovie>
<numSalle>1</numSalle>
<numCopy>2</numCopy>
<debut>2016-12-18</debut>
<fin>2016-12-30</fin>
<heure>14:50:00.000000</heure>
<archivee>0</archivee>
</projection>

Archive

<archivage>
<idMovie>252983</idMovie>
<perennite>6</perennite>
<placesVendues>120</placesVendues>
<nbrCopies>2</nbrCopies>
</archivage>

Feedback

<feedback>
<demande>
<idDemande>-612853510</idDemande>
<errors>
<error>Veuillez indiquer l'id d'un film existant</error>
<error>Le numéro de salle 5 n'exsite pas</error>

```
  </errors>
 </demande>
 <demande>
  <idDemande>-311550334</idDemande>
  <errors>
   <error>Veuillez indiquer l&apos;id d&apos;un film existant</error>
   <error>Le numéro de salle 6 n&apos;exsite pas</error>
  </errors>
 </demande>
</feedback>
```

Salle

```
<salle>
 <idSalle>6</idSalle>
 <numSalle>4</numSalle>
 <capacity>150</capacity>
</salle>
```

Séance

```
<seance>
 <idProjection>1282762075</idProjection>
 <dateSeance>2016-12-27</dateSeance>
 <nombreReservations>10</nombreReservations>
</seance>
```

Programmation

```
<programmation>
        <demande idDemande="1282762075">
                <idMovie>41524</idMovie>
                <numCopy>2</numCopy>
                <debut>2016-12-18</debut>
                <fin>2016-12-30</fin>
                <salle>1</salle>
                <heure>14:50:00</heure>
        </demande>
        <demande idDemande="-822745309">
                <idMovie>117630</idMovie>
                <numCopy>1</numCopy>
                <debut>2016-12-18</debut>
                <fin>2016-12-30</fin>
                <salle>2</salle>
                <heure>17:00:00</heure>
        </demande>
</programmation>
```

CreateLogTable.sql

Création d'une table permettant d'assurer un suivi des actions effectuées dans la base CC
Création de la procédure Log_info
Paramètres entrant : description de l'erreur, script concerné, niveau d'erreur

Process : Procédure permettant l'enregistrement d'un log

Création d'une séquence et de son trigger associé afin de gérer la génération de la clé primaire de la table Log_info.

CreaActorExt.sql

Création de la table Actors_Ext, elle comprend toutes les informations relatives aux artistes. Cette table sera utilisée avec VerifActeurs pour mettre à jour les données dans BP.

ProcReceiptMovies.sql

Procédure dont le but est de récupérer les films et copies qui sont en attentes de transfert de CB vers CC. Pour effectuer cela, la procédure va récupérer les tuples présents dans les tables de communications COM_MOVIES et COM_COPIES via le link CB_LINK. Les données étant déjà au format XML, la procédure ajoute simplement les données dans la base CC et les supprime dans les tables de communication de la base CB.

RetourCopies.sql

Création d'une procédure qui identifie les copies qui ne sont plus utilisées par une projection afin de les renvoyer. Celles-ci sont transférées dans la table COM_RETOUR_COPIES d'où CB pourra les récupérer. La procédure supprime également les projections en question.

ListingProgFiles

Création d'une table temporaire PROG_FILES qui servira à stocker les fichiers trouvés. La table supprime son contenu lors de chaque COMMIT.
Création d'une source JAVA avec en paramètre un String indiquant le répertoire dans lequel on va chercher les fichiers contenant les demandes. Chaque fichier du répertoire correspondant à la regex suivante : ^prog_\\d{4}_\\d{2}_\\d{2}_\\d+.xml$ sera ajouté dans la table PROG_FILES

AjoutProg.sql

PROCEDURE AJOUT_ALL_PROG

Paramètres entrants : chemin du répertoire contenant les différentes demandes de projections
Process : Appel Ajout_prog pour chaque fichier de demandes trouvé dans le répertoire indiqué.

PROCEDURE AJOUT_PROG

Paramètres entrants : nom du fichier contenant des demandes
Process :
Génère un document de feedback dans un fichier. Ce fichier contiendra les erreurs éventuelles liées aux demandes.
Vérifie pour chaque demande si elle est valide (date, disponibilité, …). S'il n'y a aucune erreur, un document XML sera créé afin de contenir la nouvelle projection.
Insèrtion dans la table « projections » de la projection si celle-ci est valide. Les séances correspondantes sont créées et ajoutées dans la table « séances »

PROCEDURE AJOUT_ERROR

Paramètres entrants : idDemande, message
Process : Insère dans le document XML de feedback le message d'erreur pour une demande spécifique.

PROCEDURE GET_PROG_FILES

Procédure appelant un code Java qui a pour but de récupérer tous les fichiers de demandes dans un répertoire. Les noms de fichiers sont enregistrés dans la table PROG_FILES.

FUNCTION VERIF_DISPONIBILITY

Process : Si le paramètre p_salle est spécifié, la fonction va vérifier que la salle est disponible pour une heure de début et une durée de film donnée.
Si p_copy est spécifié, la fonction va vérifier que la copie n'est pas utilisée dans une autre salle au même moment.

PROCEDURE JOB_AJOUT_ALL_PROG

Procédure destiné à un job. Celle-ci appelle la procédure AJOUT_ALL_PROG.

CreaJobAjoutProg.sql

Création d'un job qui appelle la procédure JOB_AJOUT_ALL_PROG quotidiennement.

CreaTableArchivageLogs.sql

Création d'une table de log dont chaque tuple contient un champ « date_execution ». Cette table permet de savoir lors d'un retour de copies si l'archivage a déjà été effectué pour le jour en question.
Création d'une séquence et d'un trigger pour mettre à jour automatiquement la clé primaire de la table de logs.

ArchProg.sql

Création d'une procédure ARCH_PROG qui a pour but d'archiver les projections.

Paramètre : dateTest => valeur par défaut (NULL). A utiliser uniquement pour faire des tests dans le cas où on veut modifier la date par rapport à celle de current_date.

Process : Récupère toutes les projections en cours. Une recherche va essayer de trouver une archive contenant le même film. Si elle est trouvée, on modifie ses valeurs, sinon un nouveau document XML est créé.
La procédure modifie la pérennité qui correspond au nombre total de places vendus lors des séances de la projection. Elle incrémente également de un le nombre de copies utilisées si on se trouve lors du premier jour de la projection.

CreaJobArchProg.sql

Création d'un job appelant la procédure ARCH_PROG quotidiennement. La procédure ARCH_PROG doit être exécuté quotidiennement s'il l'on veut garder la cohérence des archives.

VerifActeursTrigger.sql

Création d'un trigger sur la table Movie qui se déclenche lors d'un insertion dans la table Movies. Il récupère toutes les informations concernant les artistes du nouveau film et les envoie à BP.
Il crée un document JSON de ce format.

```
{
 "actor" :
 {
  "_id":123,
  "name" : "Alberto Dupont"
 },
 "film" :
 {
  "adult": false,
  "character": "Florent",
  "id" : 3453452
 }
}
```

Le document est récupéré par le Serveur Web dans la méthode verifacteurs du Controller APIController.php. Celui ci va vérifier et mettre à jour les données dans BP.

RechPlacePackage.sql

Ce package est principalement utilisé par l'application web. Toutes les fonctions de ce package renvoient un document XML comprenant la réponse.

FUNCTION GETHORAIRE

Récupère la liste des films qui font partie d'une programmation en ce moment.

FUNCTION GETFILMINFO

Paramètre entrant : l'id du film
Renvoie les informations du film sélectionné en paramètre.

FUNCTION GETFILMSCHEDULE

Paramètre entrant : l'id du film
Renvoie les différentes séances du film sélectionné en paramètre.

FUNCTION GETACTORS

Renvoie les liste de tous les acteurs existant

FUNCTION GETDIRECTORS

Renvoie la liste de tous les réalisateurs existant

FUNCTION GETGENRES

Renvoie la liste de tous les genres existant

FUNCTION GETPOSTER

Paramètre entrant : l'id du film
Renvoie un blob comprenant le poster du film sélectionné en paramètre

FUNCTION RECHERCHEFILM

Paramètres entrant : l'opérateur pour la popularité, popularite recherchée, opérateur pour le pérénnité, pérénnité recherchée, titre recherché, liste d'acteurs recherché, liste de genre recherché.

Recherche dans la liste des films faisant partie d'une programmation les films correspondant aux critères sélectionnés.

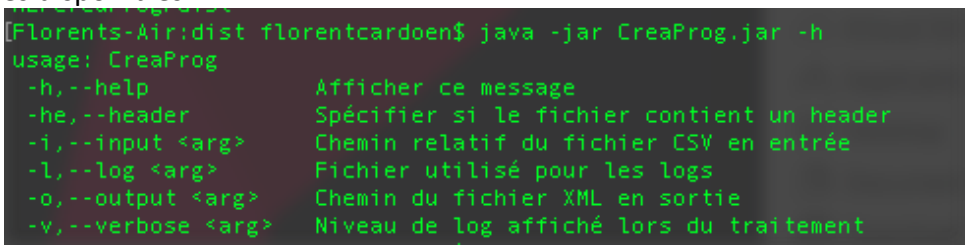
FUNCTION COMMANDEPLACE

Paramètre entrant : un document xml comprenant les commandes de tickets
Verifie la disponibilité des places. Si il y a assez de places disponible les tickets sont réservés sinon une erreur est renvoyée.

Applications Java

CreaProg

Cette application transforme un fichier de programmation CSV en XML. Celle-ci utilise apache Commons CLI pour parser les paramètres lors du lancement de l'application.
Paramètres disponibles :



CreaProg.java

Après le parsing des paramètres, l'instanciation de l'application commence avec les paramètres correspondants.
Celle-ci instancie un objet CSVParser.
Ensuite crée un document DOM en mémoire, le valide l'arbre mémoire avec le XSD correspondant et enfin écrit le résutlat dans un fichier XML si le document est valide.

CSVParser.java

Parse le fichier sélectionné et parse les informations grâce à une regex. Instancie un object de la classe Programmation.java pour chaque ligne du fichier CSV.

LogManager.java

Permet de logger dans un fichier dans un fichier ou non (en fonction des paramètres) tout ce qui s'est passé lors de l'exécution.

Programmation.java

Contient toutes les informations relatives à une programmation (debut, fin, film, copie, salle, heure, id).
Vérifie le format des informations lors de l'instanciation. En cas d'erreur, une exception "InvalidFieldException" est lancée.

VisuProg

VisuProg.java

Chaque boutons à une méthode qui permet de sélectionné les fichiers spécifiques.
Lors de l'appuie sur le bouton "Transformer", l'application charge le fichier de feedback en mémoire et le valide avec le fichier XSD.
Pour effectuer la transformation on donne en paramètre le nom du fichier de programmation correspondant au fichier de feedback car le but est de combiner en un fichier HTML les informations de la demande et le résultat de la demande.
Le fichier HTML en sortie est basé sur bootstrap.

Web

L'application web est basée sur un framework web respectant le modèle MVC. La documentation est disponible ici : https://mvc.swith.fr

Controllers

Chaque méthode publique dans un controller est une route du site web. Par exemple, la méthode affiche dans la classe ActeursController.php peut être atteinte par l'url "http://localhost/acteurs/affiche/{id}".

ActeursController.php

Gère l'affichage des informations des artistes. Ce controller communique surtout avec BP.

Index()

Permet d'afficher un formulaire pour rentrer l'id d'un artistes.

Affiche()

Récupère les informations dans BP et les affiches sur la vue.

ApiController.php

Gère une petite API REST respectant la spécification JSEND (https://labs.omniti.com/labs/jsend). Toutes les méthodes privées servent aux formatages des réponses de l'API.

Verifacteur($id)

[POST] : Met à jour les informations envoyée dans BP.
[GET] : Renvoie les informations de l'artiste donnée en paramètre.

Recherche()

[POST] : Lance la procédure recherche du Package RechPlacesPackage. Tous les paramètres sont envoyés depuis la vue du Controller RecherchesController.php. Renvoie la liste de tous les films trouvés.

Poster()

[GET] : Lance la procédure getposter du package RechPlacesPackage. Envoie le poster du film si elle est disponible sinon elle renvoie une miniature pour dire que le poster n'est pas disponible.

PanierController.php

Gère tous ce qui concerne le panier d'un utilisateur. Les paniers sont stockés dans la session niveau serveur et non dans la base de données. Ce controller utilise la classe métier App\Helper\PanierSession.php. Le système a été pensé pour être générique. Si nous voulions changer la manière de stocker les places réservées, il suffit de réécrire une classe implémentant l'interface IPanier et qui stocke les information d'une manière différente. Enfin il suffit de changer l'objet qui est instancié dans le constructeur de ce Controller.

Index()

Permet d'afficher la liste des tickets réservé sur la vue.

Add()

Permet d'ajouter au panier la séance donnée dans le POST.

Reset()

Vide le panier de l'utilisateur.

Remove($identifier, $quantite)

Permet de supprimer du panier un certain nombre de tickets du panier

Validate()

Permet de créer un document XML validé avec un DTD et de l'envoyer à la procédure commandePlace du package RechPlacePackage de la base de données CC.
Les différentes réponses sont affichées sur la vues.

RecherchesController.php

Gère la recherche de places et l'affichage des informations d'un film.

Index()

Récupère la liste des acteurs et des genres disponibles pour la recherche. La vue est composée d'un javascript qui communique en JSON avec l'API. Lorsque le JS reçoit une réponse, il crée une table avec les résultats obtenus.

Helpers

Ce namespace contient une série de classes métier qui sont utlisées dans la plupart des controllers.

OConnect

Cette classe réalise la connexion avec une base de données oracle dans son constructeurs. Elle permet aussi d'appeler facilement une fonction stockée dans la base de données.

CallFunction($funcName, array $args, bool $isBlob)

Cette fonction crée un bloc pl/sql avec le nom de la fonction et bind les différents paramètres de l'appel de la fonction aux paramètres donnés en paramètre de la fonction. Cette fonction gère également les résultats Clob.

OClient

Cette classe se situe un niveau plus haut que OConnect. Elle encapsule un Objet OConnect et map les méthodes définie dans le Package RechPlacesPackage.
Chaque méthode de cette classe passe par la méthode CallFunction de la classe OConnect.

PanierSession

Cette classe implémente l'interface IPanier et permet de gérer les réservations de tickets dans la session d'un utilisateur.
Les tickets sont stockés dans la variables superglobales $_SESSION["panier"].

Seance

Encapsule toutes les informations d'une séance. Elle convertit le document XML recu après l'appel de la fonction PL/SQL en objet Séance.
Cette classe rend la réservation de places plus simple et l'affichage des séances plus simple pour la vue.