Table des matières

Identification

Type de document : Spécification fonctionnelle et technique de l'api « Gestion bibliothèque-Livre

Emetteur : KWIZERA HUGUES TEDDY / UNICE – M2MBDS

Auteur:

Personne	Fonction	Organisme	Visa
HUGUES	Etudiant	UNICE	

Approbation:

Personne	Fonction	Organisme	Visa
Gali	Professeur	Tokidev	

Liste des versions et révisions

Version/revision	**Date**	**Objet**
1.0	**04/11/2017**	**Version initiale**

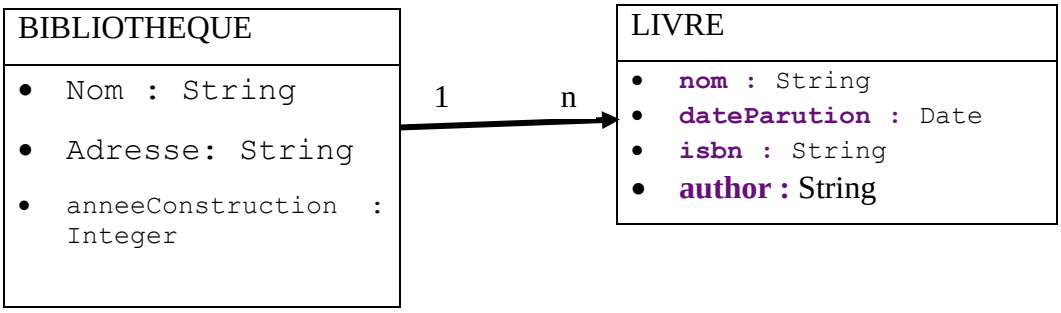
API GESTIONS BIBLIOTHEQUE_LIVRE

I. Objectif

Le but de ce travail est de mettre en place une API REST complète couvrant tous les besoins possibles sur un modèle de donnée simple, composé de deux entités liées par une composition. Les deux entités sont formées par une entité bibliothèque et Livre. Ces entités sont liées par une relation de composition.

II. Les propriétés de notre modèle

On a ce qui suit comme modèle de données :



III. Les besoins et Spécifications fonctionnels

Notre Api droit répondre aux quatre besoins essentiels de n'importe quelle modèle de donné à savoir la création, la lecture, la mise à jour et la suppression d'une entité dans la base de donnée (CRUD).

IV. Réponse fonctionnels

Pour répondre aux besoins fonctionnels de ce modèle : notre Api va comporter six méthodes :

- Deux méthodes pour la gestion de l'entité bibliothèque
- Deux méthodes pour la gestion de l'entité livre
- Deux dernières méthodes qui gèrent les ressources liées. C'est-à-dire à partir d'une bibliothèque on peut récupérer les livres qui s'y trouvent.

a. Gestion de l'entité bibliothèque

Comme c'est souligné en haut, l'entité « bibliothèque » est géré par deux méthodes. La première répond aux besoins de la lecture d'une collection des bibliothèques, l'autre va offrir à l'utilisateur de notre api la possibilité de

- récupérer une ressource bibliothèque à partir de son identifiant
- Ajouter dans la base de données une nouvelle ressource bibliothèque
- Modifier ou mettre à jour une ressource bibliothèque
- Supprimer une ressource de la base de données

b. Gestion de l'entité Livre

Même résonnement que pour l'entité bibliothèque.

c. Gestions des ressources liées

Entre les entités de notre modèle, il y a une relation de composition, ce qui nous donne la possibilité de trouver une ressource ou une collection livre à partir d'une ressource bibliothèque. Pour répondre à ce besoin, L'api « GESTIONS BIBLIO_LIVRE » va devoir mettre en place deux méthodes, l'une pour une ressource lié qui récupérer à partir d'une bibliothèque tous les livres de ce bibliothèque, et l'autre qui appliques les fonctionnalités CRUD sur un livre d'une bibliothèque.

V. Spécification techniques

Choix du Framework

L'Api « GESTIONS BIBLIO_LIVRE » va être implémenter en Grails . La version à utiliser est la 3.3.

RESSOURCE	**Verbe**	**url**	**Headers**	Params	Codes
	GET	http://127.0.0.1:8085/api/livre/	Accept : JSON X-Auth-Token	• id	• 200 OK • 400 Bad Request
	GET	http://127.0.0.1:8085/api/livres/	Accept : JSON X-Auth-Token		• 200 OK • 400 Bad Request

	POST	http://127.0.0.1:8085/api/livre/	X-Auth-Token	• Id • dateParution • isbn • bibliotheque.id	• 201 created • 400 : manque d'un des params • 500 : no biblio.id
	PUT	http://127.0.0.1:8085/api/livre/	X-Auth-Token	Un ou plusieurs dans : • Id • dateParution • isbn	• 201 created • 400 :manque de l'id du Livre
	DELETE	http://127.0.0.1:8085/api/livre/{id}	X-Auth-Token	• id	• 200 :ok • 400 : livre non existant
RESSOURCE BIBLIOTHEQUE	GET	http://127.0.0.1:8085/api/bibliotheque/	Accept : JSON X-Auth-Token	• id	• 200 OK • 400 Bad Request
	GET	http://127.0.0.1:8085/api/bibliotheques/	Accept JSON X-Auth-Token		• 200 OK • 400 Bad Request
	POST	http://127.0.0.1:8085/api/bibliotheque /	X-Auth-Token	• nom • adresse • anneeConstruction	• 201 created • 400 : manque d'un des params
	PUT	http://127.0.0.1:8085/api/bibliotheque /	X-Auth-Token	Un ou plusieurs dans : • nom • adresse • anneeConstruction	• 201 created • 400 : manque de l'id du Bibliothèque
	DELETE	http://127.0.0.1:8085/api/bibliotheque /	X-Auth-Token	• id	• 200 :ok • 400 : Biblio non existant

RESSOURCES LIEES	GET	http://127.0.0.1:8085/api/bibliotheque/{idBib}/livre/	Accept JSON X-Auth-Token	• idLivre	• 200 OK • 400 biblio ou livre inexistent • 404 : no biblio Id
	GET	http://127.0.0.1:8085/api/bibliotheque/{idBib}/livres/	Accept JSON X-Auth-Token		• 200 OK • 400 biblio inexistent ou pas de livre dans le biblio
	POST	http://127.0.0.1:8085/api/bibliotheque/{idBib}/livre/	X-Auth-Token	• IdLivre • dateParution • isbn	• 201 created • 400
	PUT	http://127.0.0.1:8085/api/bibliotheque/{idBib}/livre/	X-Auth-Token	Un ou plusieurs dans : • Id • dateParution • isbn	• 201 created • 400
	DELETE	http://127.0.0.1:8085/api/bibliotheque/{idBib}/livre/{idLiv}	X-Auth-Token	• idLivre	• 200 :ok • 400 : biblio ou livre no existant

	Verbe	Url	Parms	Body / row	code
SECURITE	POST	http://127.0.0.1:8085/api/login		{“username”: “your_username”, “password”: “your_password”}	• 200 : OK • 400 : bad request • 401 : non aurisé

	POST	http://127.0.0.1:8085/api/logout	X-Auth-Token : un token généré lors du login		

VI. Implémentation de la sécurité

Dans l'api «GESTIONS BIBLIOTHEQUE_LIVRE » , on s'est basé sur le plugin « Spring security core » et « spring security rest » pour assurer la sécurité . le plugin se base sur :

- Le Principe du token
- pour ce, le plugin limite les échanges contenant les identifiants sous la forme :
    - {"username": "your_username", "password": "your_password"}
- l'utilisateur se connecte une fois avec l'url du login (voir le tab) et recupère un token de durée limitée
- L'utilisateur utilise ensuite le « Token » pour jouer le rôle d'identifiant
- A chaque accès à une ressource le « Headers » de la requête doit contenir le token.

Different code d'erreurs :

1. 403 Forbidden : Pas de token dans le Url


```
{
   "timestamp": 1509806517038,
   "status": 403,
   "error": "Forbidden",
   "message": "Access Denied",
   "path": "/api/livre/"
}
```


2. 401 : Unauthorized : mauvais token ou expiré
3. 406 : Not Acceptable

VII. Conclusion

L’api « GESTIONS BIBLIOTHEQUE_LIVRE » a été une source d’inspiration sur le fonctionnement des « Web services », en particulier la celles mis en place grâce au Protocol REST.