



télécom
saint-étienne
école d'ingénieurs
nouvelles technologies

Java Académie #2

BOUHERROU

Nacer

MARIN

Olivier

AHMED BACHA

Abdelkrim

Février 2015

École associée
INSTITUT
Mines-Télécom

Université Jean
Monnet
SAINT-ETIENNE

Projet java académie n°2

Application Web

Descriptif de l'application

Dans un premier temps, le but de cette application est d'afficher une liste d'artistes. En cliquant sur un artiste donné, on accède à la liste de ses albums. En cliquant sur l'un des albums, on accède à la liste des musiques contenues dans cet album.

Dans un second temps, l'équipe a ajouté des formulaires pour que l'utilisateur puisse saisir les informations concernant l'une de ces 3 entités (musique, album ou artiste) et qu'elle soit persistée dans la base de données.

Formulaire permettant de d'ajouter les entités **chanson**, **album** et **artistes** :

Inscription Artiste

Code artiste :

Nom de l'artiste :

Submit

Ajout album

Code Album

Titre album

Submit

Ajout chanson



Code Chanson

Titre

Duree

Submit

L'utilisateur pourra toujours supprimer ou mettre à jour l'une de ces entités grâce aux icônes de mise à jour et de suppression.

759	Bruce Springsteen		
-----	-------------------	---	---

Remarque : La création et la mise-à-jour fonctionne pour chacune des entités du projet. En revanche, seule la suppression des chansons fonctionne.

Frameworks utilisés

Afin de simplifier le développement (ne pas réinventer la roue) et de se concentrer sur le coeur de l'application et la logique métier, nous avons utilisé des frameworks externes.

- Log4J 2.1 : pour la génération des logs de l'application et l'enregistrement dans un fichier (app.txt)

```
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-core</artifactId>
  <version>2.1</version>
</dependency>

<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-api</artifactId>
  <version>2.1</version>
</dependency>
```

- Hibernate Core: comme ORM, pour la persistance des données.

```
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-annotations</artifactId>
  <version>3.5.6-Final</version>
</dependency>
```

- Hibernate Annotation: pour le mapping des objets (entités)

```
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-annotations</artifactId>
  <version>3.5.6-Final</version>
</dependency>
```

- MySQL Connector: pour la connexion à la base de données MySQL

```
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>5.1.34</version>
</dependency>
```

- La suite Spring Framework : pour gérer l'injection de dépendance, de transaction ainsi que son architecture mvc :

```
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-context</artifactId>
  <version>4.0.7.RELEASE</version>
</dependency>

<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-web</artifactId>
  <version>4.0.7.RELEASE</version>
</dependency>

<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-webmvc</artifactId>
  <version>4.0.7.RELEASE</version>
</dependency>

<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-orm</artifactId>
  <version>4.0.7.RELEASE</version>
</dependency>
```

- JSTL : cette librairie contient l'implémentation des tags nécessaires pour l'édition d'une .jsp uniquement sous forme de balise (et non scriptlet) :

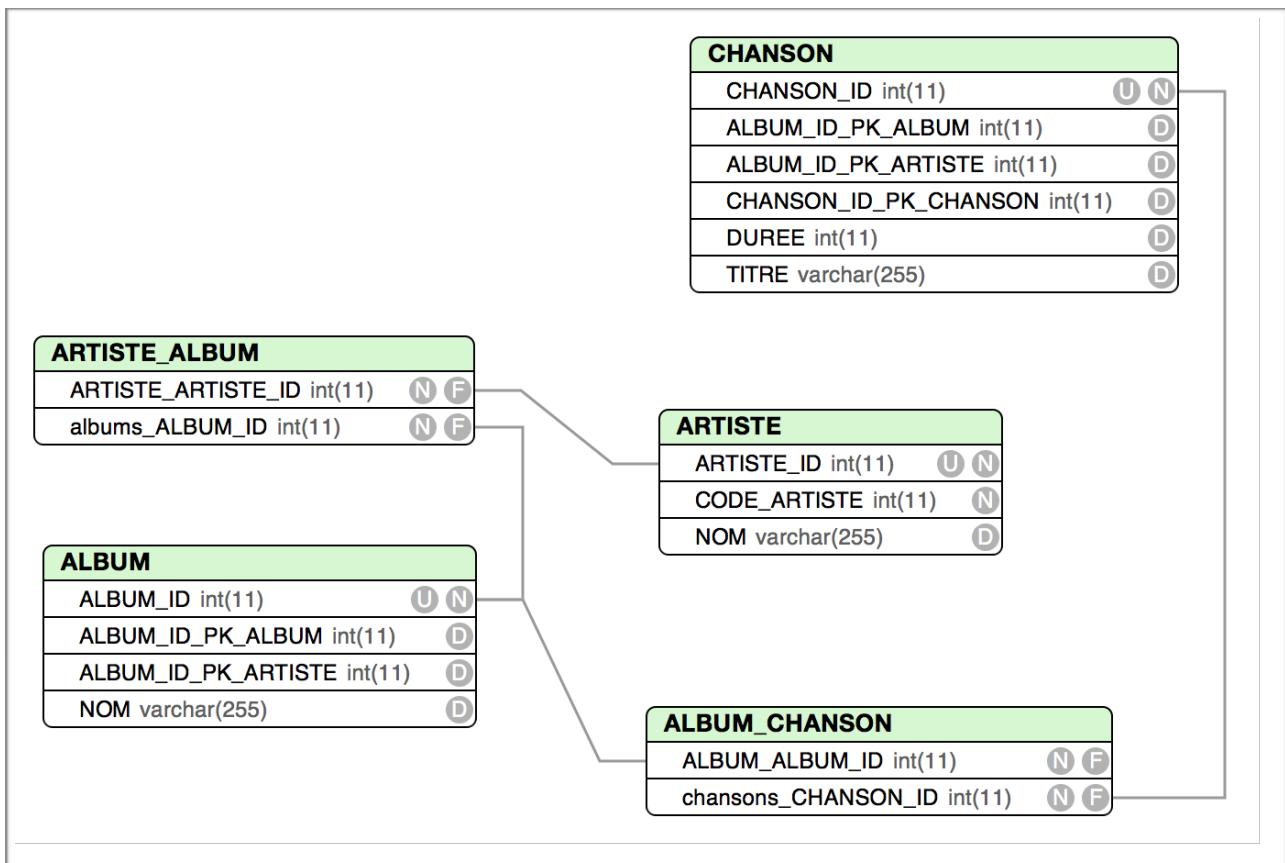
```
<dependency>
  <groupId>jstl</groupId>
  <artifactId>jstl</artifactId>
  <version>1.2</version>
</dependency>
```

Modèle de données

Pour réaliser cette application, nous avons choisi de représenter chaque objet du domaine (entités) par une table en base de données dans un premier temps, chose évidente et qui simplifie la persistance. Comme base de données, nous avons utilisé « MySQL » et connecteur java/Hibernate adéquat. On relie les tables entre elles (avec des compositions) en ajoutant une liste d'albums dans la classe **Artiste** et une liste de chansons dans la classe **Album** (on parle de classe car on mappe les objets java).

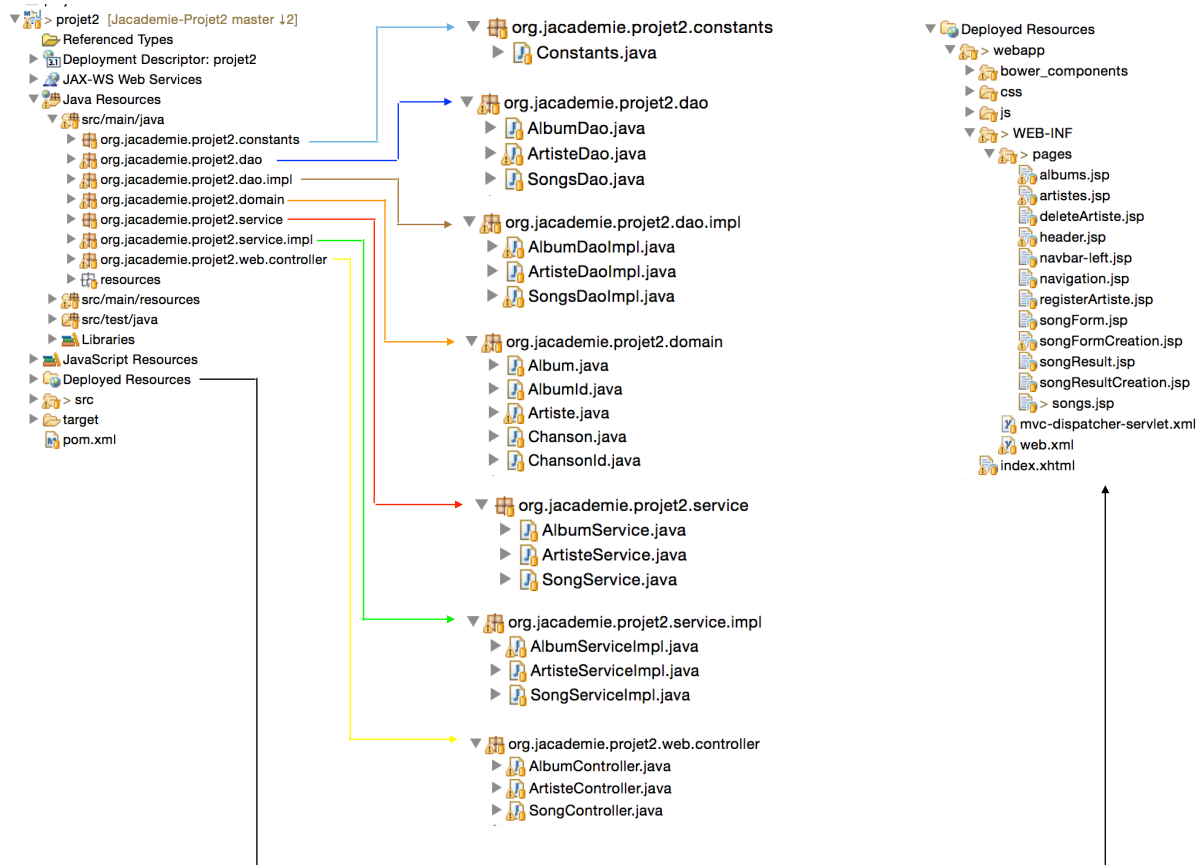
Pour garantir **l'unicité des données par artiste** (code des albums, numéros des chansons), on ajoute des clés étrangères (FOREIGN KEYS) vers les tables d'association (ARTISTE_ALBUM, ALBUM_CHANSON), et des clés primaires composées (*composite-id*) sur les tables *ALBUM* et *CHANSON*.

modèle physique de données



Pour réaliser le second projet java académie, nous avons dû reprendre la base que nous avons implémentés dans le premier projet. Cependant, la combinaison Spring / Hibernate ne permettait plus la persistance des données. Après recherche, nous avons constaté qu'il fallait ajouter une clé unique auto-générée à chaque entité de la base de données. Ceci explique l'apparition des clés **ALBUM_ID**, **ARTISTE_ID** et **CHANSON_ID** dans le diagramme de classe. Ces clés peuvent paraître redondante mais elles sont nécessaire au bon fonctionnement du projet.

Architecture de l'application et descriptif des packages



Description des package

org.jacademie.projet2.domain : ce package contient les classes. Les tables rencontrées en base de données sont ici respectivement traduites en classes métiers exploitables par le programme.

org.jacademie.projet2.dao : Les dao sont des interfaces où l'on indique uniquement la signature des méthodes.

org.jacademie.projet2.dao.impl : Ici, les méthodes des dao. sont implémentées. C'est à ce niveau que l'on injecte une « **sessionFactory** » pour créer une session et ainsi effectuer les méthodes CRUD sur les entités afin d'interagir avec la base de données.

org.jacademie.projet2.service : Dans ce package, les services ont des interfaces. Ces classes comportent donc uniquement les signatures des méthodes.

org.jacademie.projet2.service.impl : Dans ce package, on a l'implémentation des méthodes des services. C'est à ce niveau que les transactions sont assurées avec les bases de données.

org.jacademie.projet2.controller : Dans ce package, on a un contrôleur associé à chacune des entités de l'application. Chacune des méthodes implémentées dans un contrôleur charge les données affichées ou exécute les fonctionnalités proposées dans les .jsp.

Remarque 1 : Tous les formulaires de création ou de mise-à-jour utilisent bien les fonctionnalités proposées par Spring. Dans le cas d'une création, un objet est instancié et envoyé à la vue. Les champs du formulaire sont ainsi bindés aux attributs de l'objet. Ce dernier est ainsi construit puis persisté en base. Pour ce qui est de l'update, c'est le même principe sauf que l'objet est récupéré de la base de données dans un premier temps puis mis-à-jour dans un second temps.

Remarque 2 : Toute la configuration (accès à la base de données, les classes à persister et la configuration de la « `SessionFactory` ») du projet spring mvc est définie dans le fichier de configuration « `mvc-dispatcher-servlet.xml` ».

Remarque 3 : La façon dont les objets métiers persistent est définie (comme dans le projet précédent) par annotation dans chacune des classes.

Liste des fonctionnalités implantées.

- Affichage des artistes.
- Affichage des albums pour chaque albums par artistes.
- Affichage des chansons pour chaque albums.
- Création et mise-à-jour d'artistes par formulaires
- Création et mise-à-jour d'albums par formulaires
- Création et mise-à-jour de chansons par formulaires
- Suppression d'une chanson.

Configuration avant le lancement

Avant de lancer l'application, quelques étapes sont nécessaires pour finaliser la configuration.

Etape 01 : Installer un serveur MySQL si vous n'en disposez pas sur votre machine.

Etape 02 : Créer une base de données pour l'application (dans phpmyadmin ou sur MySQL)

CREATE DATABASE musique DEFAULT CHARACTER SET utf8 COLLATE utf8_unicode_ci;

Etape 03 : Dans le répertoire « **Deployed Resources/webapp/WEB-INF** », il faut modifier le paramétrage de connexion à la base de données MySQL (utilisateur, mot de passe, url de connexion avec port de MySQL).

```
<property name="hibernate.connection.url">jdbc:mysql://localhost:8889/musique</property>
```

```
<property name="hibernate.connection.username">yourUsername</property>
```

```
<property name="hibernate.connection.password">yourPassword</property>
```

Etape 04 : Après avoir ouvert le projet sur Eclipse, faire un « Maven - Update Project » pour télécharger les dépendances (frameworks) en local. Après avoir déployer l'application sur votre serveur GlassFish, veuillez-vous rendre à l'url suivante :

<http://localhost:8080/projet2/Artistes.do>

En vous remerciant !

L'équipe