# Collecte des données (Datawarehouse)

Partie Arnaud : reverse geo-coding, API…

# Ordonnancement des données

Partie Hakim : Webservices, fichiers csv…

# Stockage et exploitation

Partie Maxime : Récupération des données, stockage en base de données

## Modèle de données

Nous avons décidé de mettre en place une base de données relationnelle. Cette base de données repose sur l’existence de deux tables (cela va évoluer lors des prochains lots).

### Table Beaches

|  |  |  |
| --- | --- | --- |
| Champ | Description | Type |
| Id | Identifiant unique et non nul, valeur incrémentale | int(10) et clé primaire |
| beach\_name | Nom de la plage | string(50) |
| Latitude | Valeur de la latitude au format +/- 00.000000 | float |
| Longitude | Valeur de la longitude au format +/- 00.000000 | Float |
| Address | Adresse postale de la plage | string (150) |

Par exemple, pour la plage de Deauville :

|  |  |
| --- | --- |
| Champ | Valeur |
| Id | 0000000001 |
| beach\_name | Plage de Deauville |
| Latitude | 49.357000 |
| Longitude | 00.069950 |
| Address | Promenade des Planches, 14800 Deauville |

### Table Users

|  |  |  |
| --- | --- | --- |
| Champ | Description | Type |
| Id | Identifiant unique pour chaque utilisateur | int(11) et clé primaire |
| Login | Identifiant | string(50) |
| Password | Mot de passe | string(250) |
| Nom | Nom de famille | string(50) |
| Prenom | Prénom | string(50) |
| Birthday | Date d’anniversaire | Date |
| Email | Adresse mail | string(150) |

## Mise en place de la base de données

La base de données tourne actuellement sous phpMyAdmin lancé à l’aide de Wampserver. Chaque table est créée à l’aide d’un script SQL qui peut être réutilisé au cas où l’on veut réinitialiser les données.

### Beaches

|  |
| --- |
| USE holysearch ;  CREATE TABLE IF NOT EXISTS beaches  (  id int(10) NOT NULL AUTO\_INCREMENT,  beach\_name VARCHAR(50) NOT NULL,  latitude FLOAT NOT NULL,  longitude FLOAT NOT NULL,  address VARCHAR(150) NOT NULL,  PRIMARY KEY (id)  ) ; |

### Users

|  |
| --- |
| USE holysearch;  CREATE TABLE IF NOT EXISTS users  (  id int(11) NOT NULL AUTO\_INCREMENT,  login VARCHAR(50) NOT NULL UNIQUE,  password VARCHAR(250) NOT NULL,  nom VARCHAR(50) NOT NULL,  prenom VARCHAR(50),  birthday DATE,  email VARCHAR(150) NOT NULL UNIQUE,  PRIMARY KEY (id)  ) ; |

## Connexion de la BDD au serveur d’applications

Le serveur d’application est un Tomcat. Pour qu’il puisse exploiter nos données, des objets sont créés dans l’architecture déjà mise en place.

### Schéma général de l’exploitation des données

Ce schéma est générique et décrit la manière dont une donnée est récupérée dans le cas où un utilisateur effectue une action nécessitant l’exécution d’une requête sql sur la table BEACHES.

Controleur

Objet beachBeanTO

Recherche de l’utilisateur

Mapping entre l’objet java et l’objet TO avec la classe Mapper Utils

Création d’un Transfer Object (TO)

Objet beachBean

Requête SQL

Création de l’objet avec la classe Beach

Formulaire

Mapper Utils

Service

Les formulaires sont des classes permettant d’assurer les interactions entre l’utilisateur et le contrôleur.

### Recherche d’un utilisateur (Use case)

Chaque plage est stockée en base de données. Du côté java, chaque plage est instanciée par la classe Beach.

Dans notre architecture il s’agit de l’objet bean. Le fichier source 🡪 Beach.java

#### Récupération en base

Le contrôleur fait appel à des méthodes des classes de Service qui vont exécuter les requêtes SQL nécessaires à la recherche de l’utilisateur.

#### Résultat de la requête

Le résultat de la requête est retourné par la classe de Service adéquate et stockée dans l’objet bean.

#### Mapping

Pour être exploitable par le formulaire, l’objet beach nécessite quelques adaptations.

C’est pour cela que le controller instancie un objet beach de classe Transfer Object (TO).

Un mapping est effectué entre le bean et le TO à l’aide des méthodes de la classe MapperUtils.

#### Envoi du résultat à l’utilisateur

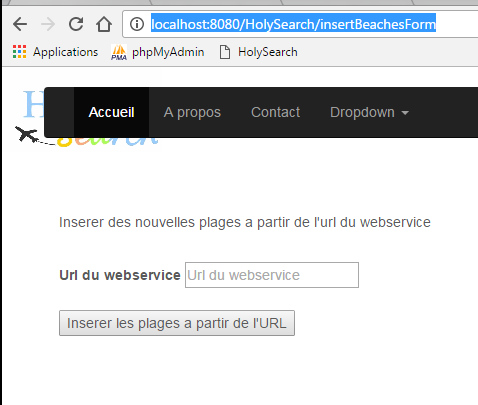
Une fois le résultat récupéré par l’objet beachTO, celui-ci est soumis par le Controleur vers le Formulaire adéquat avant de renvoyer les données qui vont être affichées à l’utilisateur. Le rôle du formulaire est surtout de décrire les données, l’affichage étant ensuite géré par des jsp.

## Insertion des données en base

Il faut se rendre sur la page principale de HolySearch et ajouter insertBeaches de cette manière :

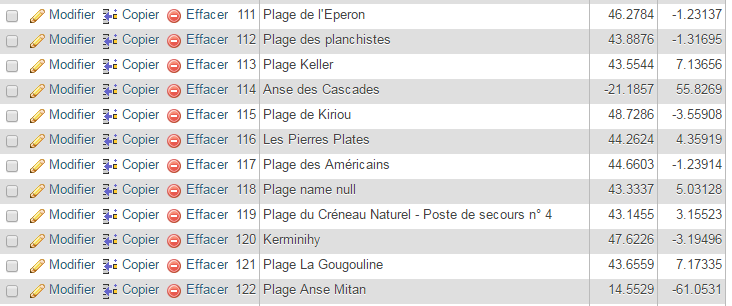
localhost:8080/HolySearch/insertBeachesForm

La page suivante s’affiche :



Insérer l’url du webservice. Cette procédure pourra être répétée chaque fois que l’on veut insérer des nouvelles données dans une table.

Pour les plages nous obtenons le résultat suivant en base (voir exemple ci-dessous) :

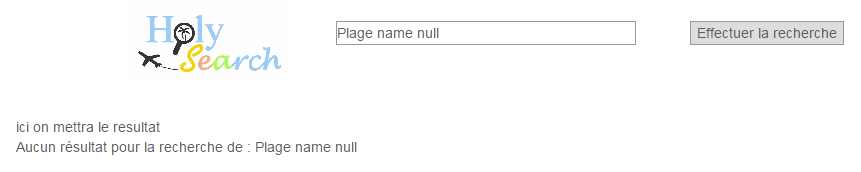


Chacune de ces plages est retournée par le moteur de recherche dès lors qu’un utilisateur en saisit le nom tel qu’il est affiché en base, à l’exception des plages nommées *Plage name null).*

*Recherche fructueuse :*

**

Recherche en échec :

**

Plusieurs plages portent ce nom car le webservice qui les récupère n’a pas tous les noms de plage. *Plage name null* est un nom donné par défaut lors de l’insertion des données en base lorsqu’il n’y a pas de nom à insérer pour une plage donnée.

La méthode qui exécute la requête de sélection par le nom ne retourne que des résultats dont le nom est unique. Comme il existe plusieurs plages sans nom, il ne les sélectionne pas.

Un moyen de remédier à cela est de modifier la méthode que l’on applique aux résultats récupérés par la requête de sélection afin d’autoriser l’affichage du résultat même en cas de *Plage name null*.

Nota bene : Cette procédure ne sera peut-être pas définitive, mais en attendant, elle permet facilement d’injecter des données en base et de les rendre exploitables par l’utilisateur.