Le CRUD 100% AJAX et JSON dans une architecture MVC

Par : Abdel YEZZA, Ph.D
Date : octobre 2015


Version PDF de l'article


Télécharger le code source

Table des matières

Introduction

Le CRUD acronyme signifiant (C : Create , R: Read ou Retrieve U : Update et D : Delete (différent de Remove), est utilisé dans tout langage de programmation possédant ces quatre opérations standards qui opèrent sur des objets, des données peu importe la forme, le format et leur conteneur. Il peut s'agir par exemple d'instances (objets créés à partir de classes ou modèles) d'un composant ou d'un jeu d'enregistrements d'une base de données ou tout simplement un fichier système (texte, hexa, binaire ou autres). Les quatre opérations s'imposent intrinsèquement par elles-mêmes et par la nature même du paradigme de la programmation OO (Orienté Objet) en étant des instances vivantes de classes (Create) subissant des modifications (Update) et des consultations (Read), et finiront par mourir (Delete).

Cet atticle s'adresse à une poulation ayant déjà une bonne base sur les différent concepts abordés et par conséquent il est foculasé sur le comment et non pas sur le quoi. Son but ne consiste pas à expliquer en détails ce que c'est le CRUD et tous les concepts associés ou utilisés, mais plutôt, comme son titre l'indique, son objectif est de montrer comment réaliser le CRUD dans un langage OO comme JAVA Le même exercice peut être réalisé dans n'importe quel autre langage OO de dernière génération comme C#, PHP ou autres.

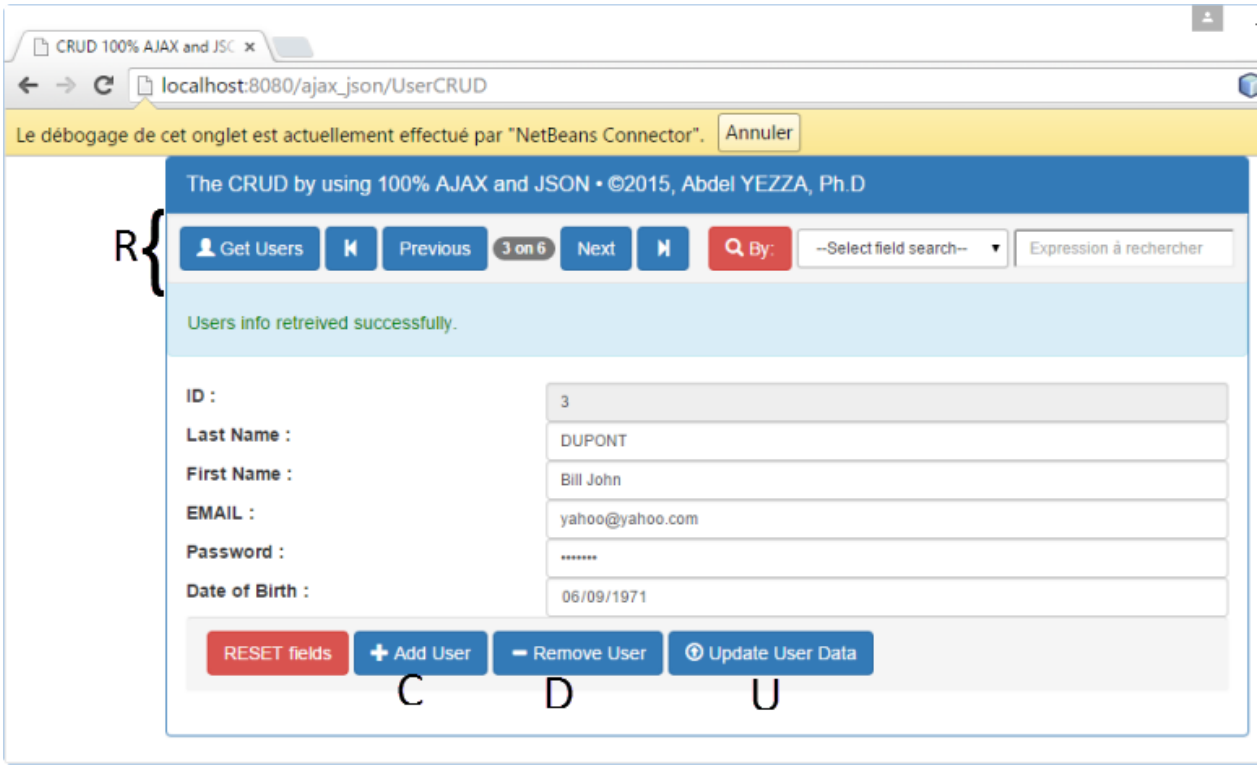
Il s'agit plus précisémment de montre à travers d'une application simple de type WEB développée en JAVA (JEE : Java Entreprise Edition. Pour le moment je ne donnerai pas plus de détails et je vous invite à lire le chapitre suivant pour découvrir tous les sujets technologiques utilisés dans cette application.

L'application

Sont abordés dans l'application accompagnant cet article, les technologies suivantes :

- Comment créer une application WEB de type JEE utilisant le modèle MVC (Model, View, Controller) : Nous allons voir comment sont réellement matérialisées les trois couches composant le modèle MVC (couche M, couche V et la couche C). Dans le cas d'une application utilisant des données provenant par exemple d'une source de données stockées dans une base de données d'un SGBD ou autres (fichiers systèmes, XML, JSON etc.), les bonnes pratiques exigent que la couche C (Controler) ne doit pas communiquer directement avec la source de données, mais passe par une couche intermédiaire dite DAO (Data Access Object)
- Comment utiliser dans la couche de présentation (couche V) le Framework qui a le vent dans les voiles ces dernières années BOOTSTRAP au sein d'une application JEE WEB.
- Comment utiliser AJAX dans une application WEB afin de ne pas recharger une vue (page WEB) entièrement, mais uniquement la partie qui doit être modifiée suite à un appel d'un composant serveur par un composant client via un évènement (click, change, etc.). La technique utilisée par AJAX contribue à l'expérience positive de l'utilisateur final.
- Comment utiliser le langage d'expression (EL : Expression Langage) par la Taglib JSP (Java Server Pages) utilisée dans la construction des vues.
- Comment contrôler et valider les données envoyées par les clients au moment de la saisie (côté client) et au moment de la réception côté serveur. Un contrôle au niveau du client avant que la requête soit partie inutilement et un contrôle à l'arrivée avant d'être traitée par les composants serveur.

Sans plus tarder, voici le résultat en image de l'application WEB formée d'une page unique réalisant les quatre opérations du CRUD, mais aussi en bonus l'opération de recherche selon plusieurs critères.



Cette application vous montre Comment :

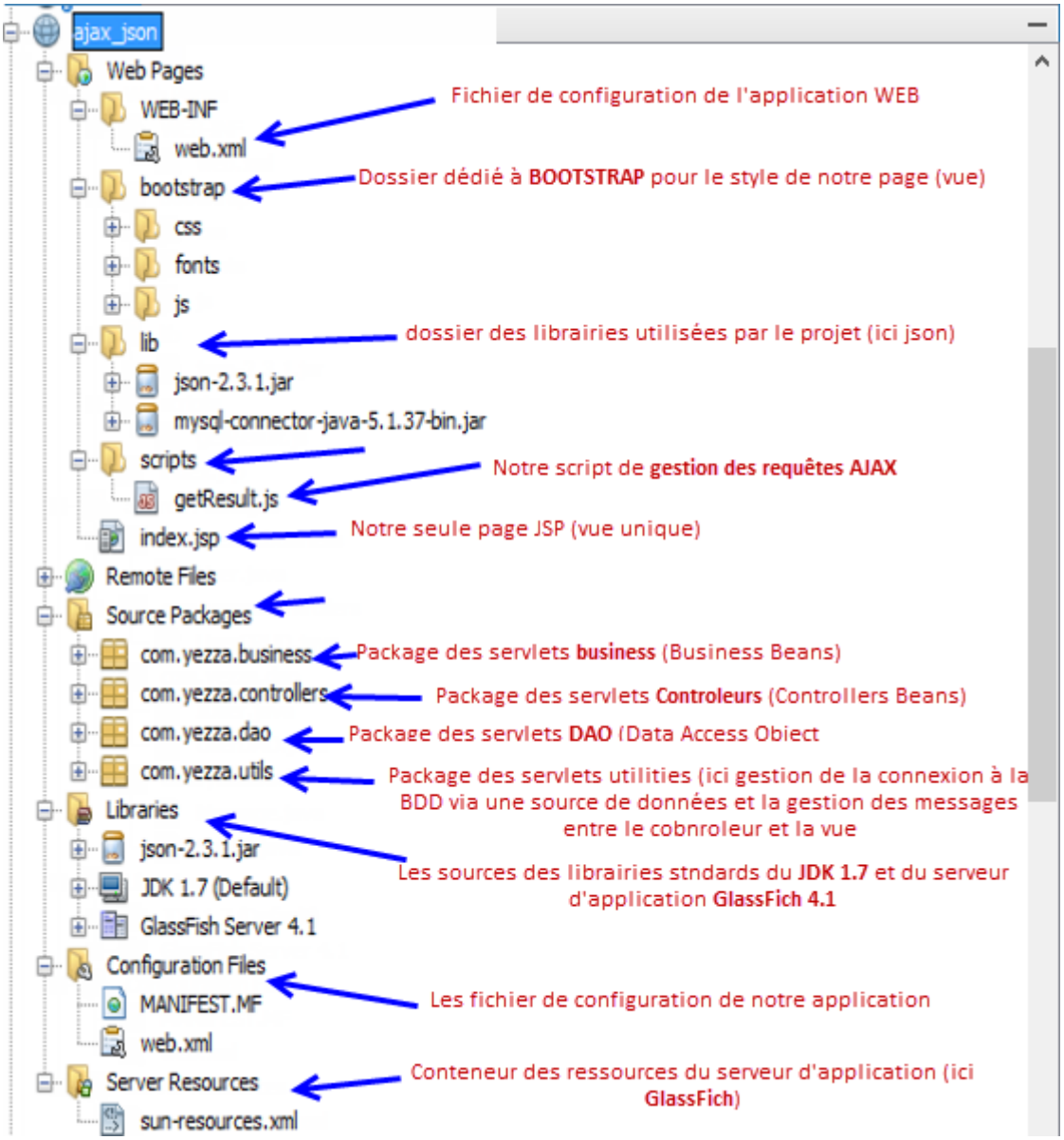
- réaliser une application WEB ayant un IHM "Responsive", autrement dit, toutes les parties de l'IHM se repositionnent en horizontal ou en vertical selon la taille de l'écran (le device) et leurs positions relatives à la page, qu'il s'agisse d'un PC de bureau, d'une tablette, voir même d'un smartphone.
- réaliser une barre de navigation dans un formulaire sans utiliser un Framework, mais uniquement AJAX et les composants serveur de l'application afin d'afficher les enregistrements selon l'action envoyée par AJAX (Get Users, Next, Previous, First et Last) sans passer par des Framework dans cette catégorie.
- réaliser une barre de recherche simple selon des critères sur les champs d'une table d'une base de données. Des recherches plus complexes doivent utiliser des Framework plus appropriés comme Spring, Hibernate, Solr etc.
- afficher des messages d'une manière dynamique dans la même page sans la recharger entièrement.
- réaliser une barre matérialisant le CRUD (Get Users et la barre de navigation = R, Add User = C, Remove User = D et Update User = U)

L'environnement de développement utilisé

Bien que l'environnement de développement l'importe peu, il est néanmoins important de le définir dès le début de n'importe quel projet de cette nature afin de bien définir les technologies, les techniques et les frameworks à utiliser. Toutes ces briques concernent :

- L'IDE (Integrated Development Environment) à utiliser : dans notre application, l'IDE Netbeans a été utilisé
- Le langage de programmation : dans notre application, il s'agit de JEE (WEB application), côté serveur avec des vues en JSP (Java Server Pages) et côté client Javascript avec AJAX
- Les Framework de programmation Serveur utilisés : Il peut s'agir de Framework intégrés dans l'IDE en tant que Plugins ou ajoutés en plus. Dans notre application, aucun Framework supplémentaire côté serveur n'a été utilisé. Du côté du client (Views, pages JSP), on a utilisé le Framework AJAX, json-23.1 de Google (appelé GSON)
- Les Framework de programmation Client utilisés : Le BOOTSTRAP comme indiqué ci-dessus.
- Les données : l'application utilise une base de données MySql formée d'une table unique (pour la construire, un fichier de script "db_ajax_json.sql" a été inclus dans le projet à exécuter par importation dans MySql Workbench par exemple ou directement dans la console MySql pour les amateurs des consoles de commandes.
- Le serveur d'application WEB : On a utilisé GlassFish intégré d'office dans Netbeans. N'importe quel autre serveur d'application open source peut être utilisé comme Tomcat, JBoss etc.

Afin de matérialiser ces différentes briques, voici en image les parties du projet segmentées tel qu'affiché dans l'explorateur projet de Netbeans :



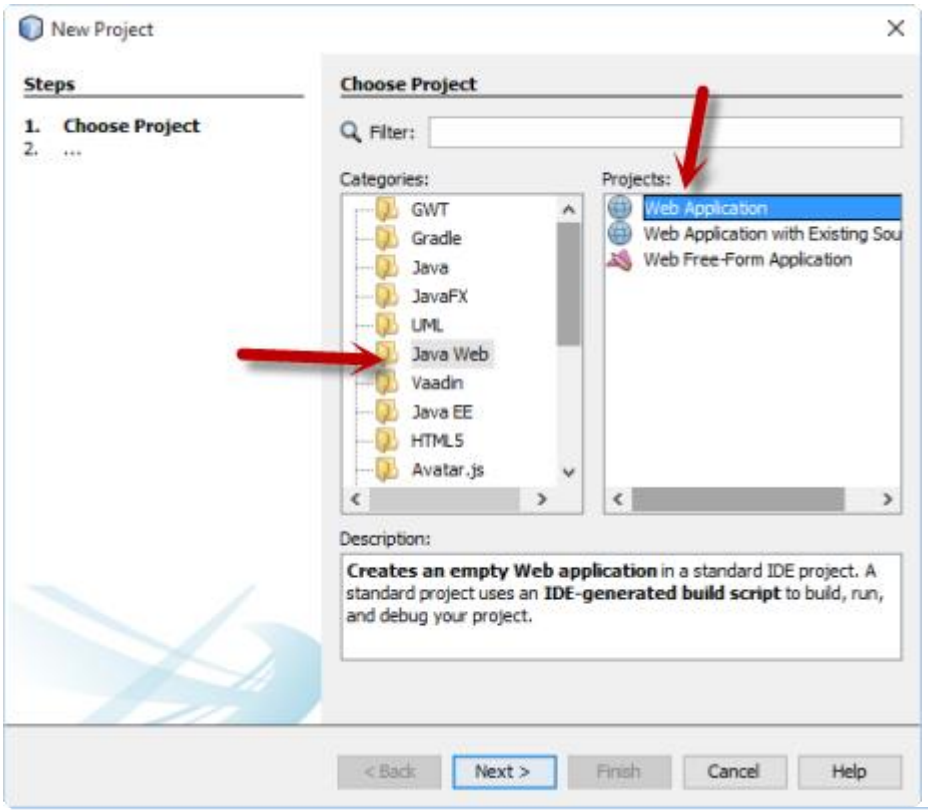
Si vous optez pour un autre IDE open source comme ECLIPSE, il est tout à fait possible d'éditer le code source accompagnant cet article dans ce dernier. On a utilisé le format de données JSON afin de réaliser la récupération des données véhiculées par AJAX. Il est aussi possible d'utiliser un autre format différent comme XML via des modifications très légères du projet. D'ailleurs, une fonction Javascript est déjà incluse dans le projet pour effectuer la récupération des données au format XML.

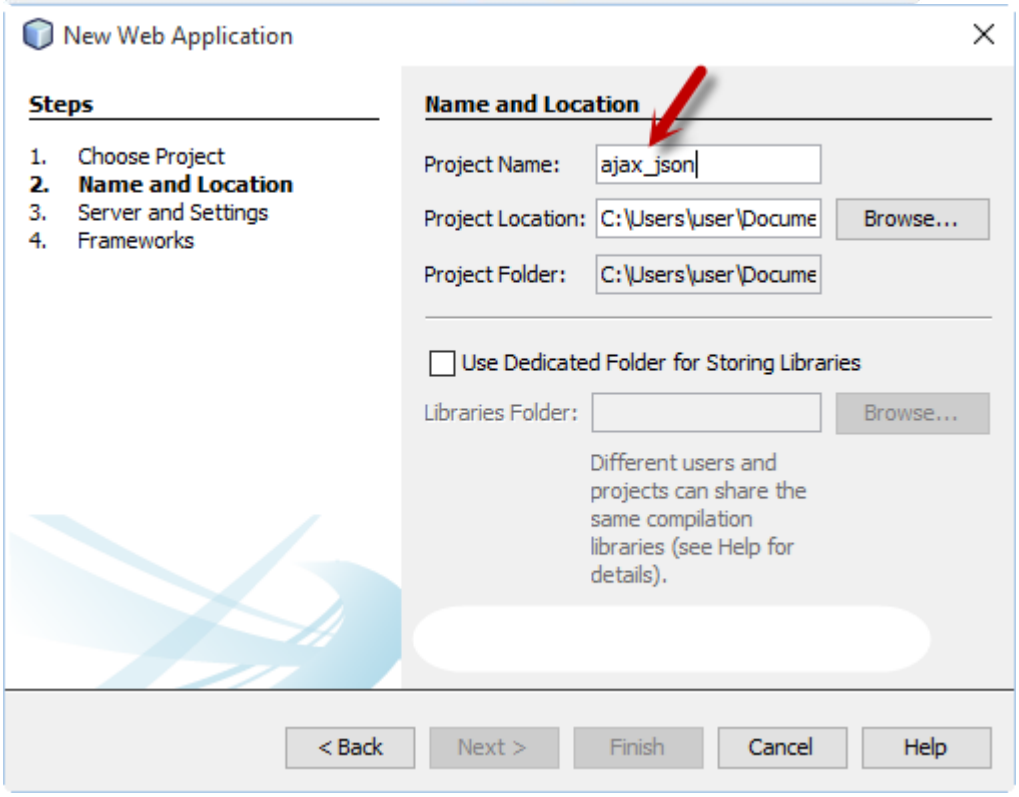
Ci-dessous, je vous explique pas à pas les étapes à suivre pour créer le projet.

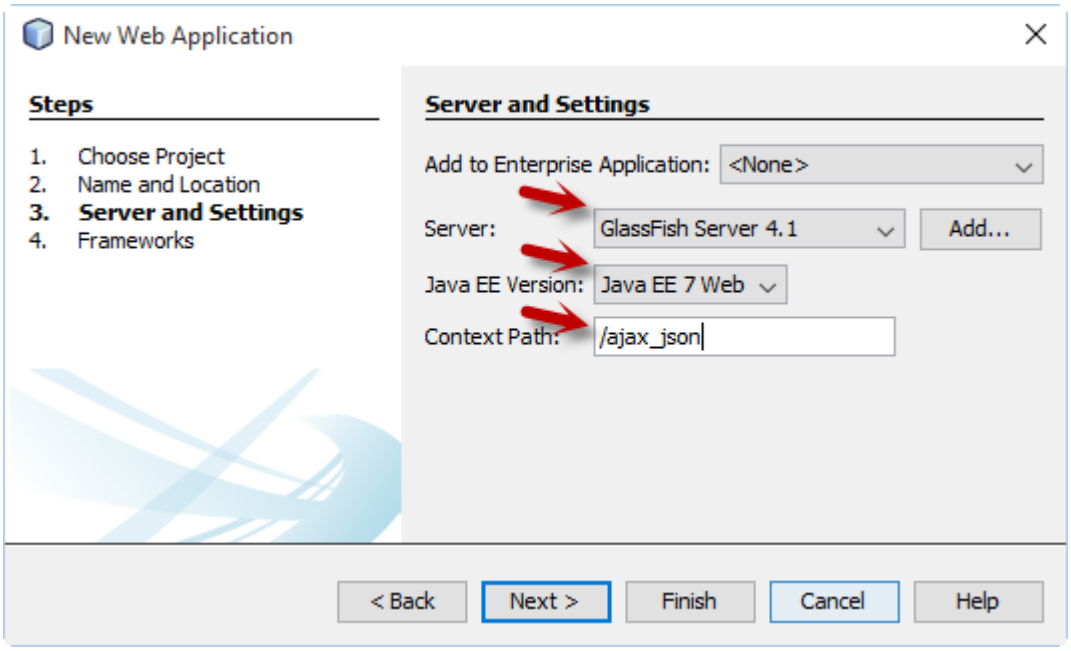
Le projet

Après avoir cliqué sur le menu File->New Project, suivez les étapes de l'assistant de création de projets telles qu'indiquées ci-contre :

A noter qu'aucun Framework ne sera utilisé pour ce projet. A noter aussi que le projet utilise le JDK 7 (Java Devlopment Kit) une librairie dédiée aux développeurs d'application JEE, ou toute autre version compatible de préférence 6 ou plus.


New Project
Steps
1. Choose Project
2. ...
Choose Project
Filter:
Categories:
GWT
Gradle
Java
JavaFX
UML
Java Web
Vaadin
Java EE
HTML5
Avatar.js
Projects:
Web Application
Web Application with Existing Sou
Web Free-Form Application
Description:
Creates an empty Web application in a standard IDE project. A standard project uses an IDE-generated build script to build, run, and debug your project.
< Back
Next >
Finish
Cancel
Help


New Web Application
Steps
1. Choose Project
2. Name and Location
3. Server and Settings
4. Frameworks
Name and Location
Project Name: ajax_json
Project Location: C:\Users\user\Docume
Browse...
Project Folder: C:\Users\user\Docume
Use Dedicated Folder for Storing Libraries
Libraries Folder:
Browse...
Different users and projects can share the same compilation libraries (see Help for details).
< Back
Next >
Finish
Cancel
Help



Le squelette de votre projets est créé avec des éléments par défaut déjà présents comme la page de démarrage "index.htm" à supprimer, le fichier de MANIFEST.MF qui sert à configurer le fichier archive (WAR : WEB ARchive) généré représentant le package de l'application à distribuer, déployer ou inclure dans une application Entreprise comprenant plusieurs composants.

La vue (Le V de MVC)

Le fichier index.html sera remplacé par un fichier JSP "index.jsp" représentant l'unique page WEB de notre application. Comme illustré ci-contre. Pour cela cliquer sur File->New File Puis sélectionner WEB puis JSP. Cette page est le seul élément qui matérialise le V dans l'architecture de type MVC. Attention à ne pas oublier d'indiquer le ou les pages de démarrage de l'application soit dans le fichier de configuration de l'application web.xml soit dans les propriétés du projet sous la rubrique Run en y indurant la nouvelle page. Dans notre cas, comme c'est notre unique page, on peut tout à fait omettre d'indiquer la page de démarrage, mais ce n'est pas une bonne pratique. Pour cela, vous ajouter manuellement les lignes suivantes dans web.xml ou l'indiquer à l'assistant lors de la création de la page jsp.

```
<welcome-file-list>
<welcome-file>
index.jsp
</welcome-file>
</welcome-file-list>
```

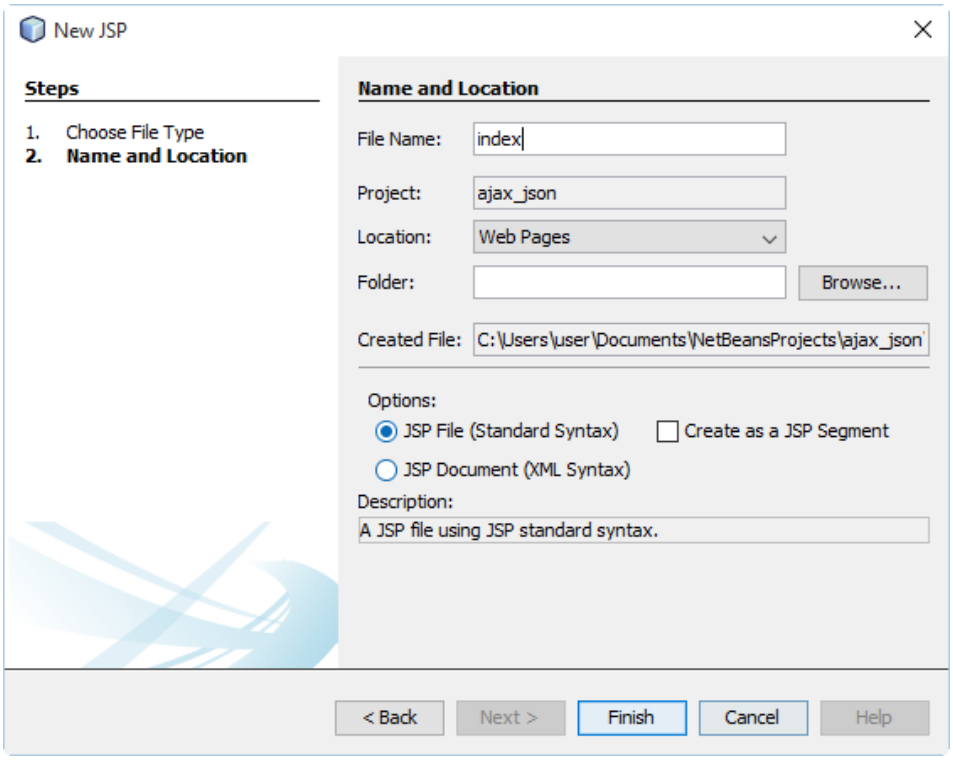
JSON

Afin d'utiliser JSON comme format de flux de données entre le client (via AJAX) et le serveur, nous avons besoin d'une librairie JSON comme celle de Google par exemple. Pour cela on peut créer un dossier lib sous le dossier web présenté par WEB Pages dans l'explorateur NetBeans ! Puis y mettre toutes les librairies Java que nous voulons utiliser. Dans notre cas, il s'agit de deux librairie, celle pour JSON et celle pour se connecter à la source de données MySql (voir ci-dessous). Voir la figure ci-contre.

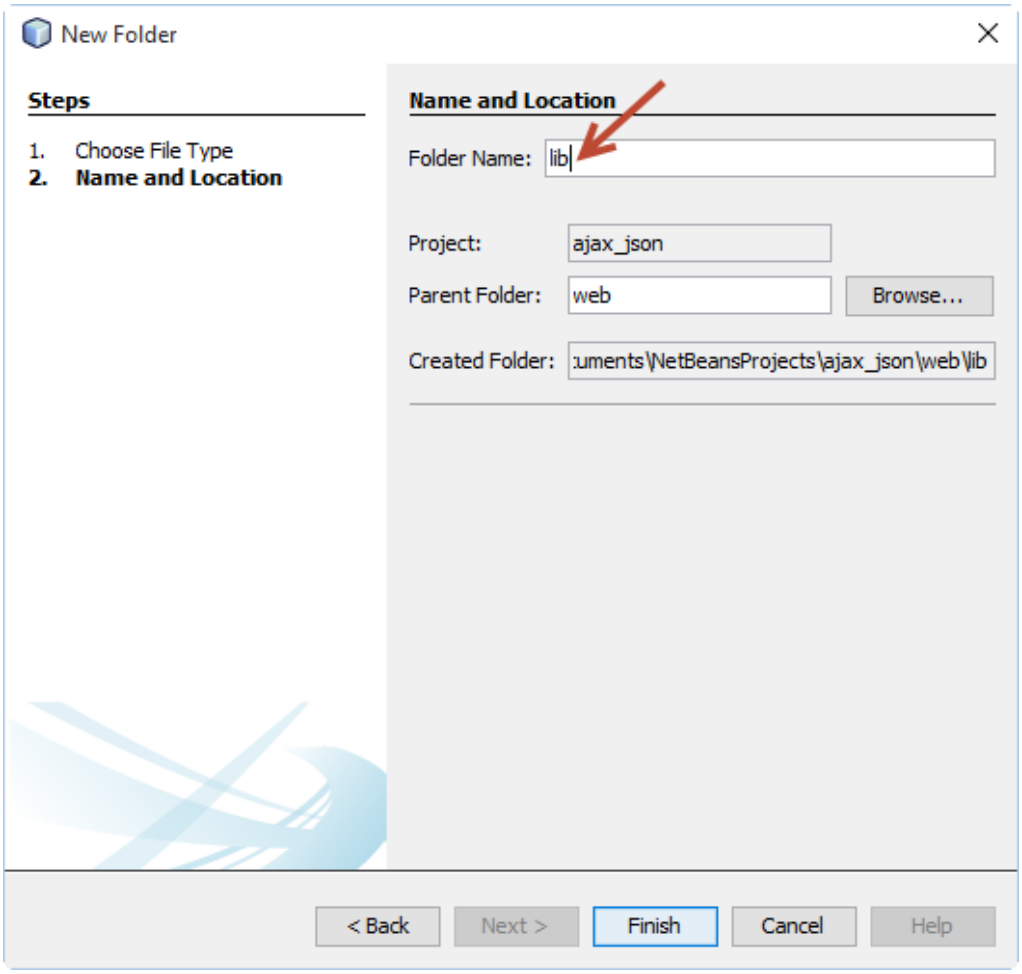
AJAX

L'utilisation de la technique proposée par AJAX a un double usage. Le premier usage est pour ne recharger dans une page que le DELTA devant être modifié suite à une requête envoyée au serveur par le client. La deuxième utilisation est dans un but de recevoir la réponse du serveur dans un flux sous un format bien défini, en l'occurrence du JSON, comme il peut être sous format XML, binaire, BLOB ou tout autre format standard ou même personnalisé. Pour cela, nous ajoutons un dossier scripts dédié à Javascript dans lequel un script appelé getResult.js est créé et qui aura la délégation du client pour communiquer avec les composants serveur en utilisant AJAX comme moyen de communication (à ne pas confondre avec le protocole standard du WEB qu'est le HTTP) et le format JSON comme flux de données.

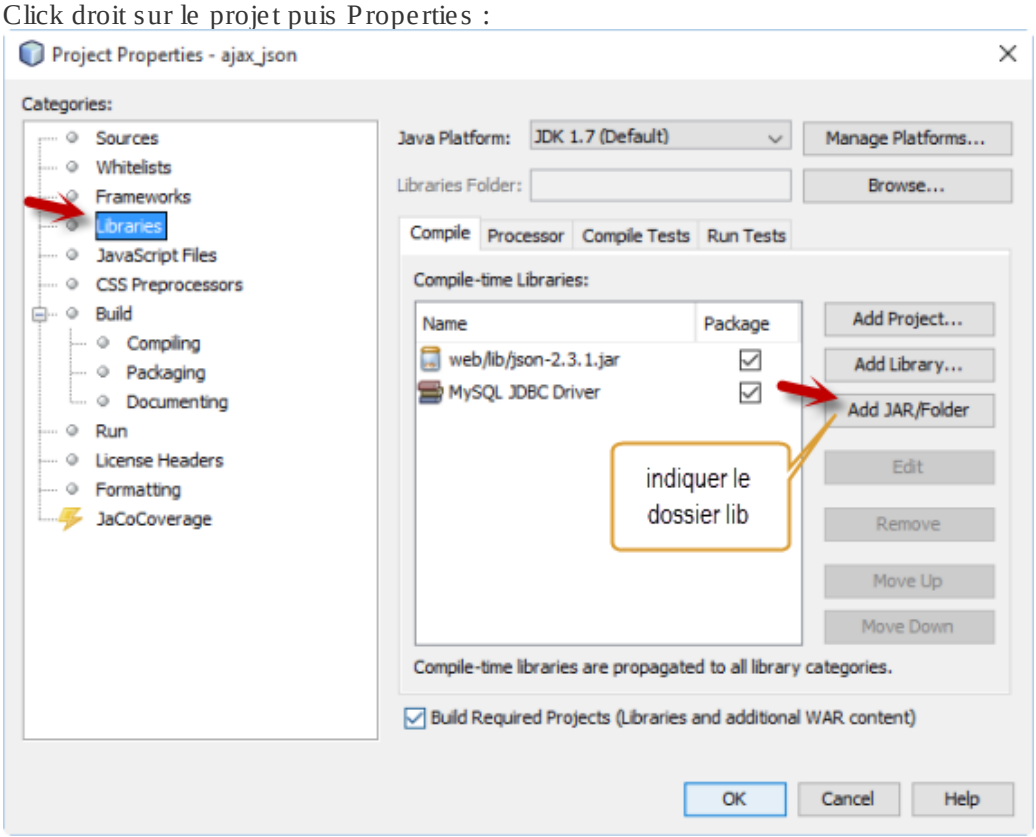
File -> New File -> WEB -> JSP

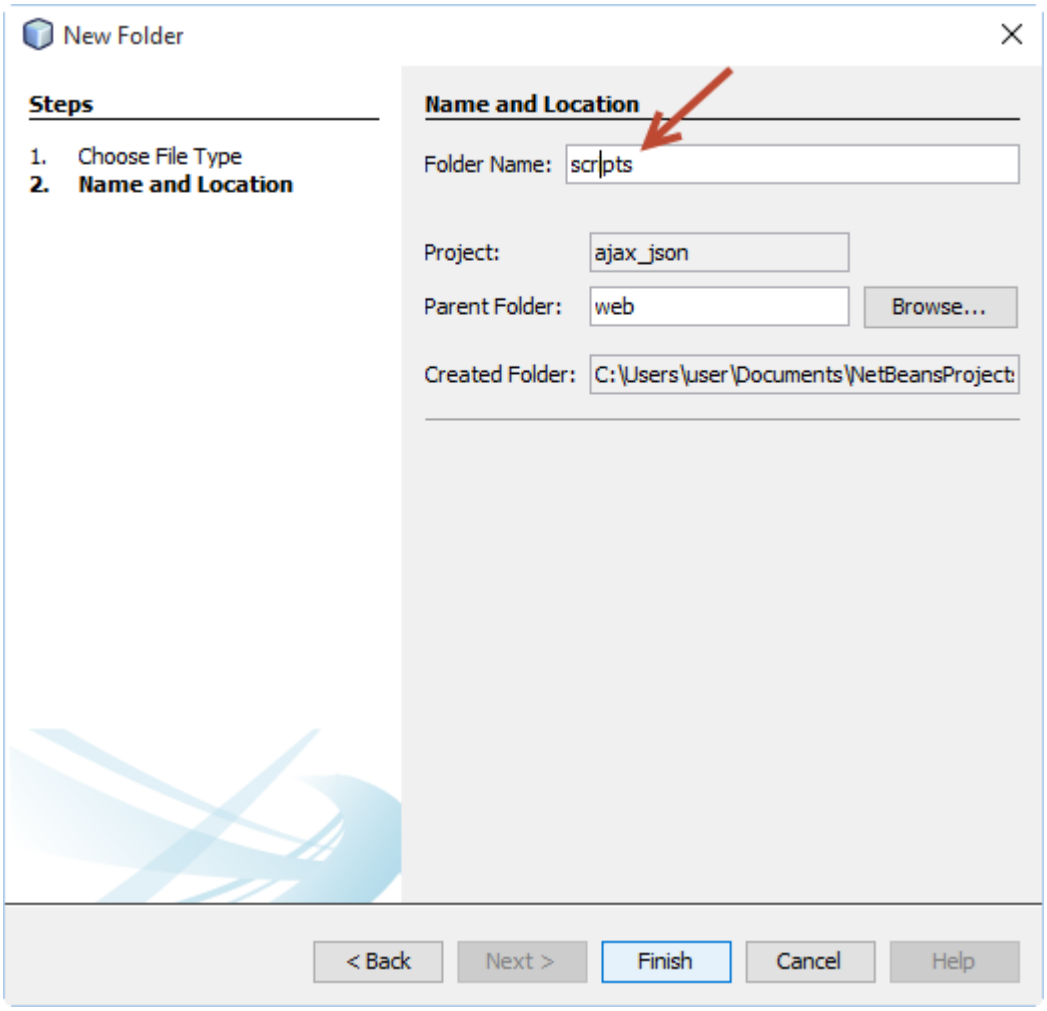


File -> New file -> Other -> Folder



Click droit sur le projet puis Properties :





BOOTSTRAP

Afin d'utiliser le Framework BOOTSTRAP dans notre page, on crée un dossier bootstrap dans lequel nous créons toutes l'arborescence appropriées contenant principalement (que vous pouvez télécharger sur le site de bootstrap) :

- La feuille de style BOOTSTRAP
- Le fichier javascript utilisé par bootstrap

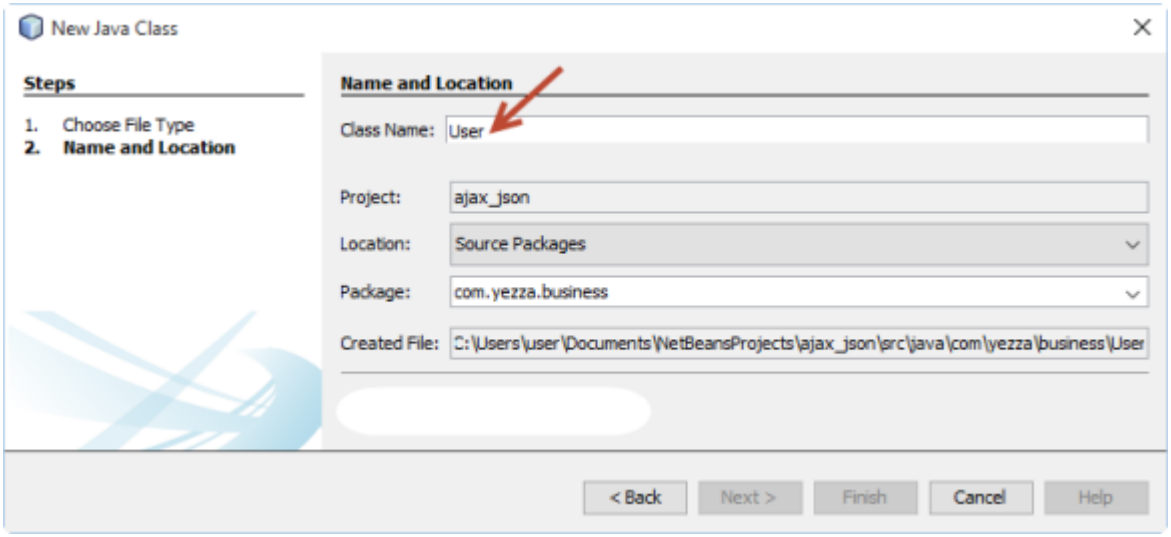
MySql JDBC Driver

Afin de pouvoir utiliser une source de données MySql, nous aurons besoins aussi du pilote (DRIVER) approprié qui va nous permettre de communiquer avec la base de données MySql via son connecteur JDBC (Java DataBase Connector). Celui-ci peut être téléchargé sur le site http://dev.mysql.com/downloads/connector/j/. Il est préférable d'inclure toutes les librairies directement avec le dossier du projet au lieu du poste de développement afin de lever toute dépendance dans le cas de changement du poste. Voir la figure ci-contre :

Les composants Business (le M de Model)

Les composants métier (business) sont rassemblés dans un package unique com.yezza.business comme indiqué dans la figure au début de cet article. Pour ce faire :

1. commencer par ajouter un package appelé com.yezza.business sous le conteneur Source Packages (représentant le dossier src du projet Netbeans). Voir la figure ci-contre.
2. Puis ajouter dans le package ainsi créé la classe métier (Bean)User qui représente un utilisateur le seul objet métier manipulé dans cette application. Voir la figure ci-contre.



Les composants C (le C de Controller)

Les composants Contrôleur sont rassemblés dans un package unique com.yezza.controllers. Dans ce package nous ajoutons un contrôleur unique appelé UserCRUD qui doit prendre en charge l'ensemble des requêtes provenant de notre unique page client index.jsp. Comme son nom l'indique, il doit être capable :

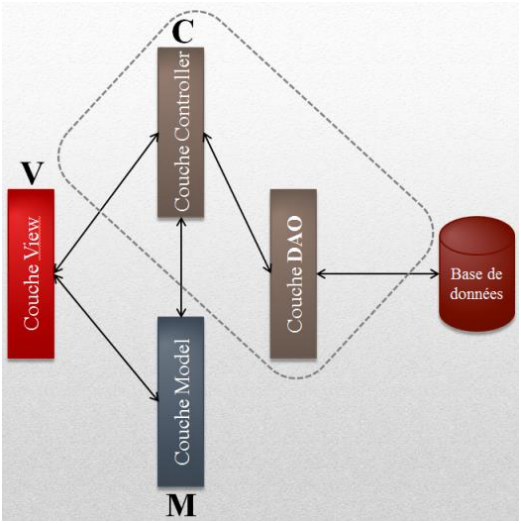
- D'effectuer les opérations du CRUD (Créer un utilisateur, Afficher un ou plusieurs utilisateurs, Mettre à jour un utilisateur et Supprimer un utilisateur)
- Et en plus, il effectue une recherche d'utilisateurs selon des critères différents par propriété (ID, nom, prénom, Email ou date de naissance).

Les composants supplémentaires

Comme il a été évoqué au début de cet article, une application WEB doit être fondée sur une architecture répartie sur plusieurs couches coopératives avec la contrainte que chaque couche ne communique qu'avec ses couches voisines directes. La coopération signifie que l'on peut créer des instances de la couche voisine pour récupérer ou fournir des données ou des objets sans pour autant s'approprier du rôle de l'autre. Dans notre application, on a déjà défini les couches V (View), C (Controller) et M (Model). Et pourtant, il subsiste une couche supplémentaire qui va s'introduire dans le C, car il est recommandé que les contrôleurs ne communique pas directement avec les sources des données et doivent passer par une couche dédiée à ce rôle appelée la DAO (Data Access Object). La figure ci-contre illustre ce principe. En résumé :

- La vue envoie des requêtes au controleur pour recevoir des réponses. Elle utilise aussi les objets du modèle afin de récupérer les données (propriétés) issues des objets métiers.
- Le contrôleur reçoit des requêtes de la vue et lui envoie des réponses dans la même vue ou dans une autre vue. Il utilise aussi le modèle business pour construire les objets métier qui contiennent toute la logique et les règles qui s'y rattachent. Il peut tout à fait communiquer avec la base de données, mais il est fortement recommandé de déléguer cette fonction à une couche intermédiaire appelée la DAO. Voir ci-dessous pour le package dédié à la DAO.

- Le modèle est responsable d'instancier les objets métiers, les manipuler en appliquant la logique définie (fonctions métier) et les règles imposées. Il instantie les objets soit à partir du contrôleur ou fournit des propriétés d'un objet à partir de la vue.



Les composants DAO

Le package com.yezza.dao est dédié à la couche DAO communiquant entre la base de données (via une source de données à définir, voir ci-dessous) et le contrôleur UserCRUD. Dans ce package, on a créé :

- Une classe UserDAO qui doit implémenter l'ensemble des interfaes entre le contrôleur et la base de données.
- Afin de réaliser cet interfaçage, un contrat entre les deux parties doit être établi. Ce contrat est matérialisé par une classe de type Interface que l'on a créé dans le même package précédent appelé IUser. Cette interface indique toutes les méthodes que la classe UserDAO doit implémenter afin de respecter le contrat. Voici un extrait de cettte interface :

```
public interface IUser {
    public boolean addUser(User pUser, AtomicReference<String> errorMsg);
    public boolean delUser(User pUser, AtomicReference<String> errorMsg);
    public boolean updateUser(User pInUser, AtomicReference<String> errorMsg);
    public String getAllUsers();    // get JSON represntation of all users
    public ResultSet getAllUsersResultSet();    // get the resultset of all users
    public String getUserById(int pId);  // get JSON represntation of users with pId serach
    public String getUsersByName(String pName);     // get JSON represntation of users with pNa
me serach
    public String getUsersByPrenom(String pPrenom); // get JSON represntation of users with pPr
enom serach
    public String getUserByEmail(String pEmail); // get JSON represntation of users with pEmail
serach
}
```

Les composants transverses

Tous les comopsants transverses pouvant être utilisés dans des contextes différents sont rassemblé dans un package unique appelé com.yezza.utils. Dans ce package on trouve deux classes :

- la classe Message qui a le rôle de construire les messages avec le contenu du message, son code et la couleur du fond qui sera utilisée dans la vue en fonction du code (succès, avertissement ou erreur fatale).
- la classe SqlUtil dont le rôle est de gérer l'objet Connection en récupérant l'instance unique de la connexion à la source de données de type (MySql) utilisée par l'application. L'objet Connection doit être unique, est construit une seule et unique fois lors du premier appel et le même objet est réutlisé à chaque appel suivant. C'est l'un des principes "du design pattern" connu sous le nom de "SINGLETON". Son code est illustré ci-contre. Au premier appel de la calsse, elle crée l'instance unique et aux appels suivants elle ne fait que fournir l'instance déjà créée.

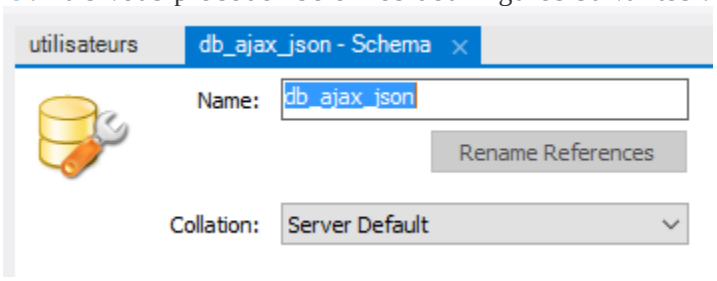
```
public class SqlUtil {

    private static DataSource dataSource;

    /**
     *
     * @return une connexion ouverte a partir du DataSource
     */
    public static Connection getConnection(){

        try {

            InitialContext context = new InitialContext();
            dataSource = (DataSource)context.lookup("jdbc/ds_ajax_json");

            return dataSource.getConnection();

        } catch (NamingException | SQLException ex) {
            Logger.getLogger(SqlUtil.class.getName()).log(Level.SEVERE, null, ex);
        }

        return null;
    }

}
```

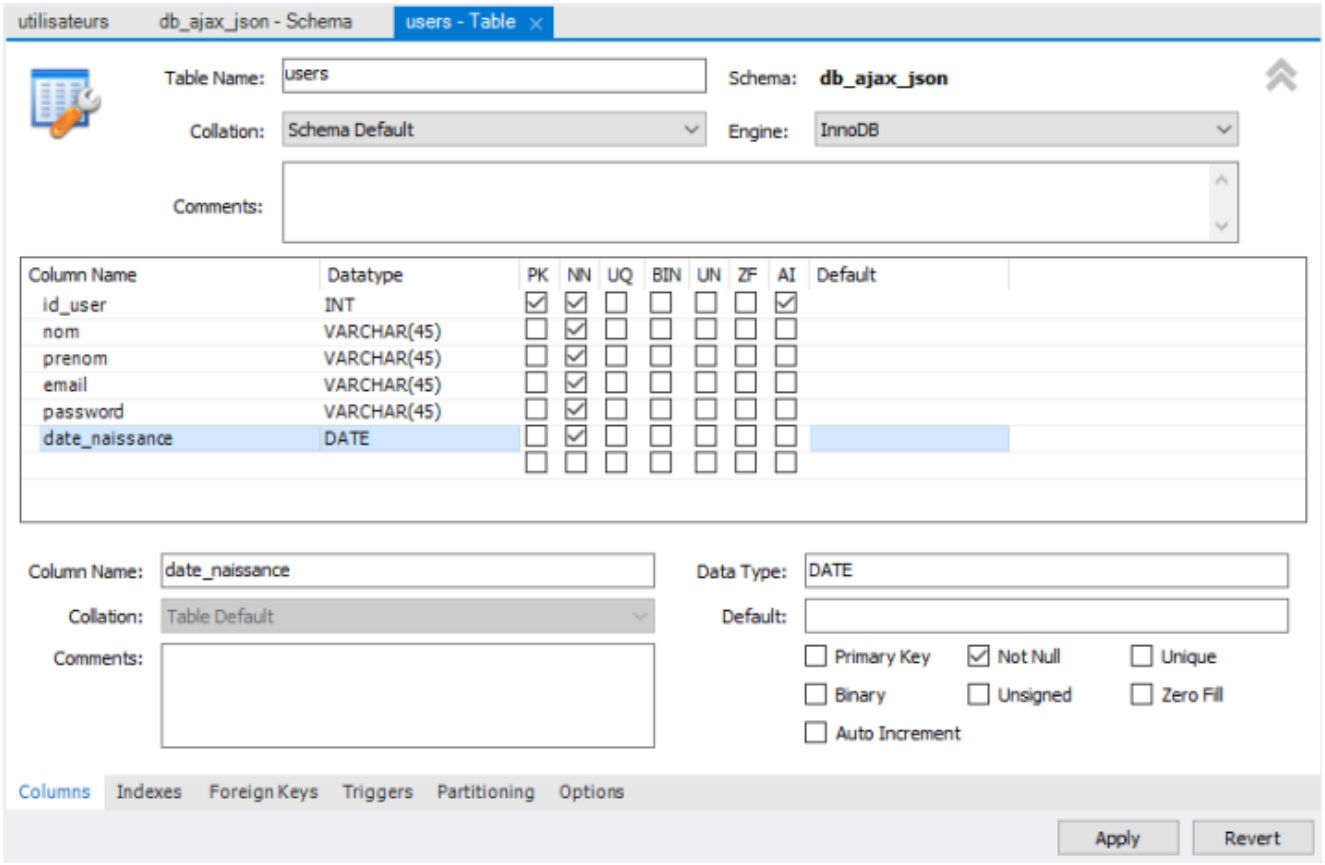
L'unique méthode statique pouvant être appellée pour récupéter l'objet **Connection**

Notre unique source de données non visible à l'extérieur

La base de données

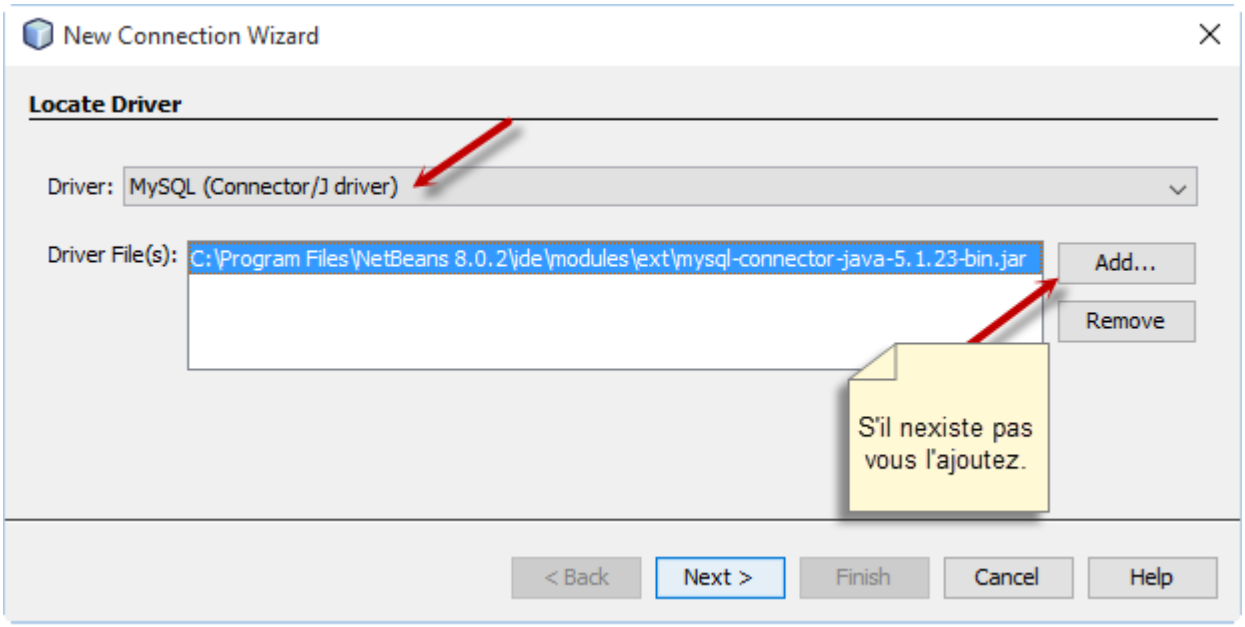
Les données de l'application sont stockées dans une base de données MySql db_ajax_json formée d'une table unique users. Comme on a déjà inclus le Driver de MySql, vous devez avoir dans l'IDE Netbeans dans l'onglet Services une entrée intitulée : MySql Server at localhost:3306... (voir la figure ci-contre). Netbeans vient avec un certain nombre de drivers qui se trouvent sous l'entrée Drivers. Vous avez aussi la possibilité d'en rajouter d'autres en fonction du SGBD que votre application utilise.
Afin de créer la base de données db_ajax_json, vous pouvez utiliser par exemple MySql Workbench téléchargeable ici. Puis vous procédez selon les deux figures suivantes :

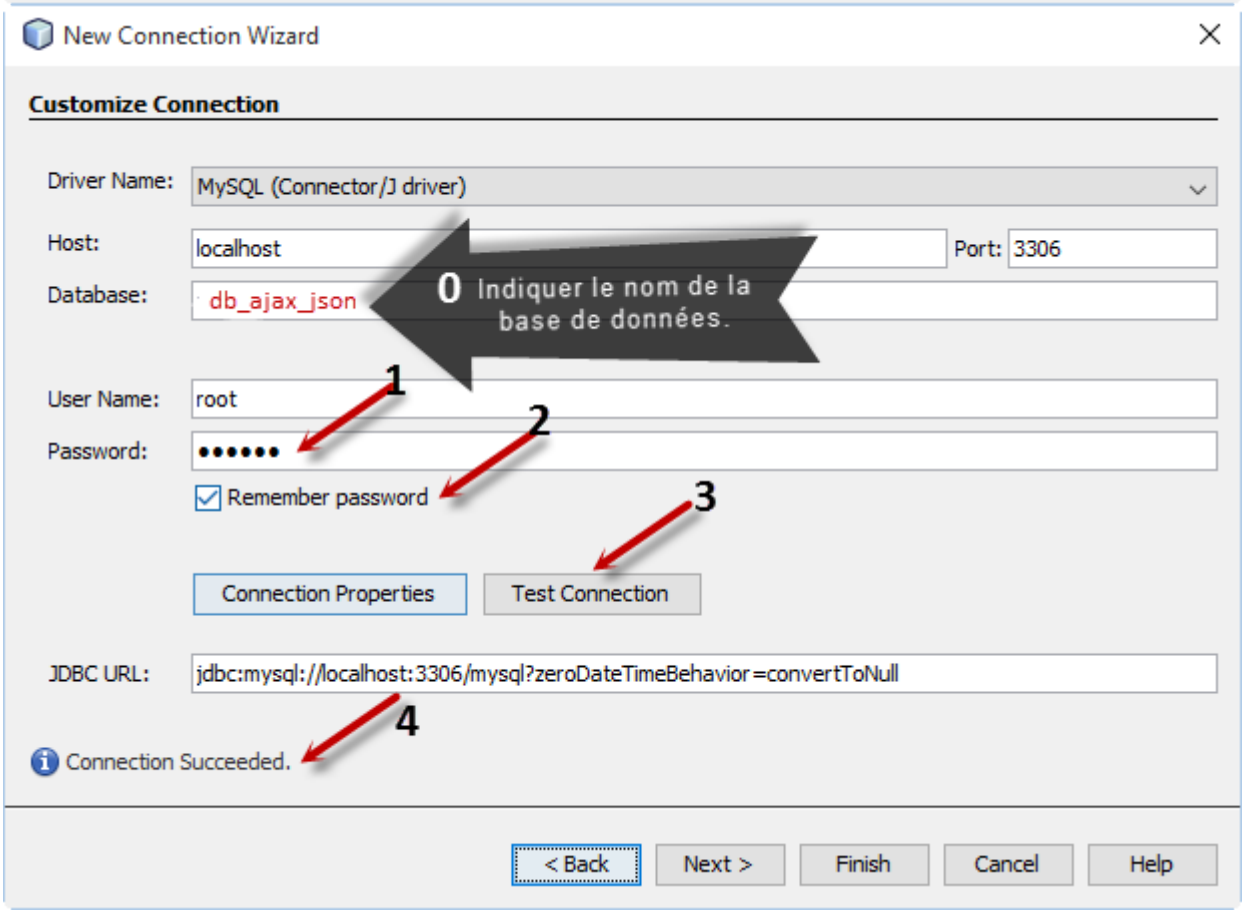




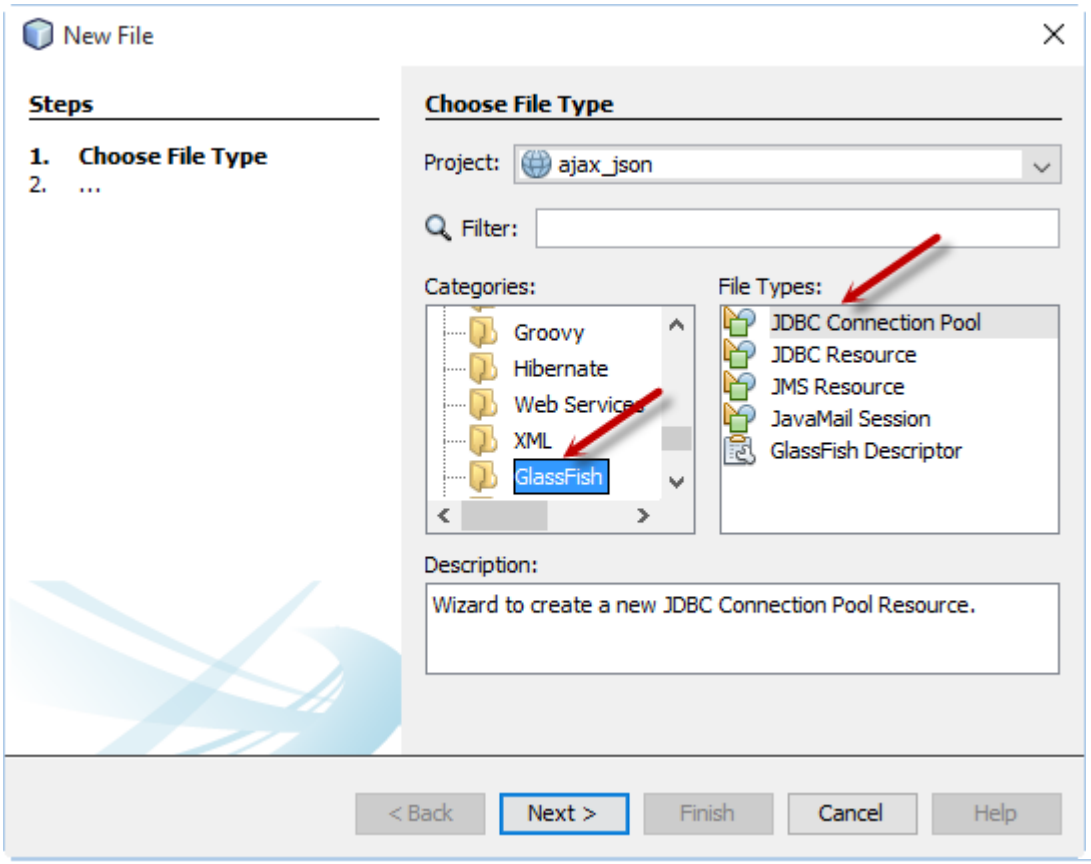
Afin que l'application puisse se connecter à la base de données, vous avez besoin de 3 éléments :

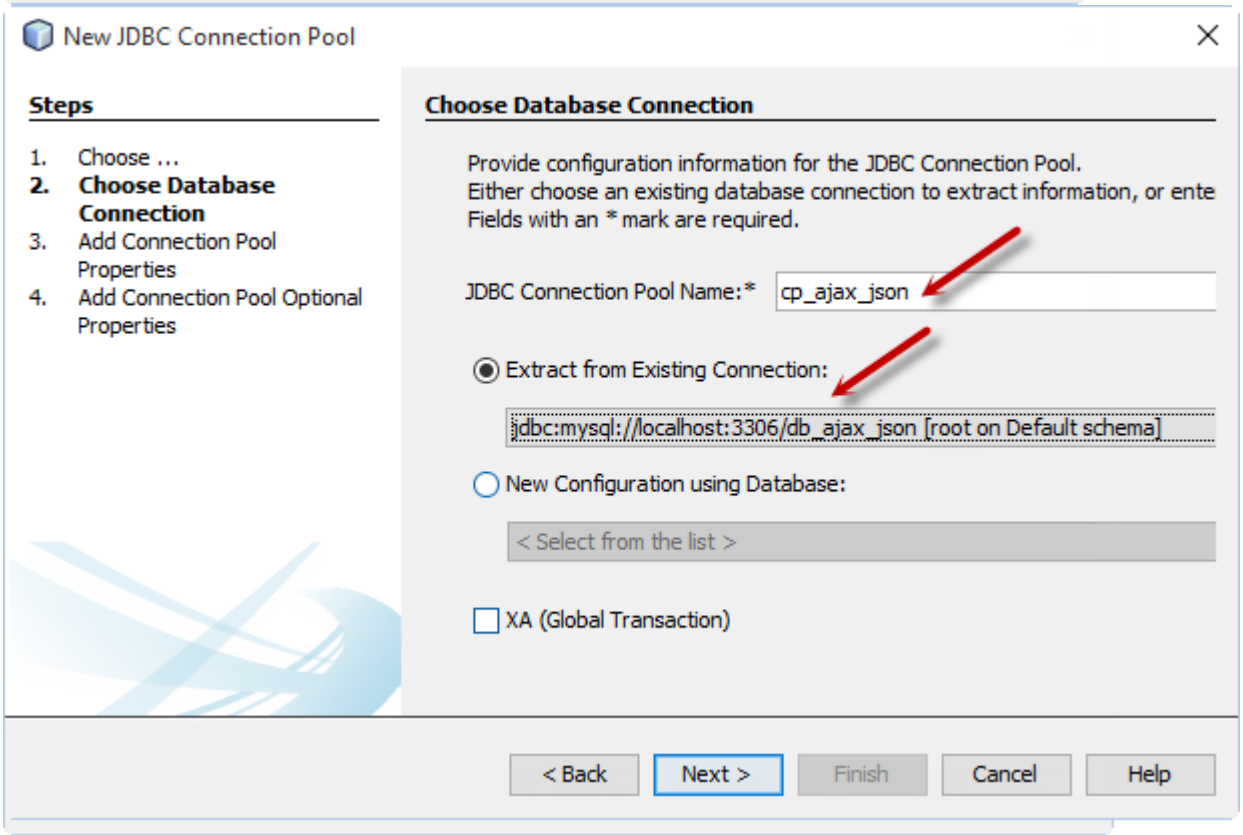
1. Au moins une connexion à la base de données : Pour cela, vous vous positionnez sur l'onglet Services (Menu : Window --> Services), click droit sur l'entrée Databases puis New Connetion comme indiqué ci-dessous :

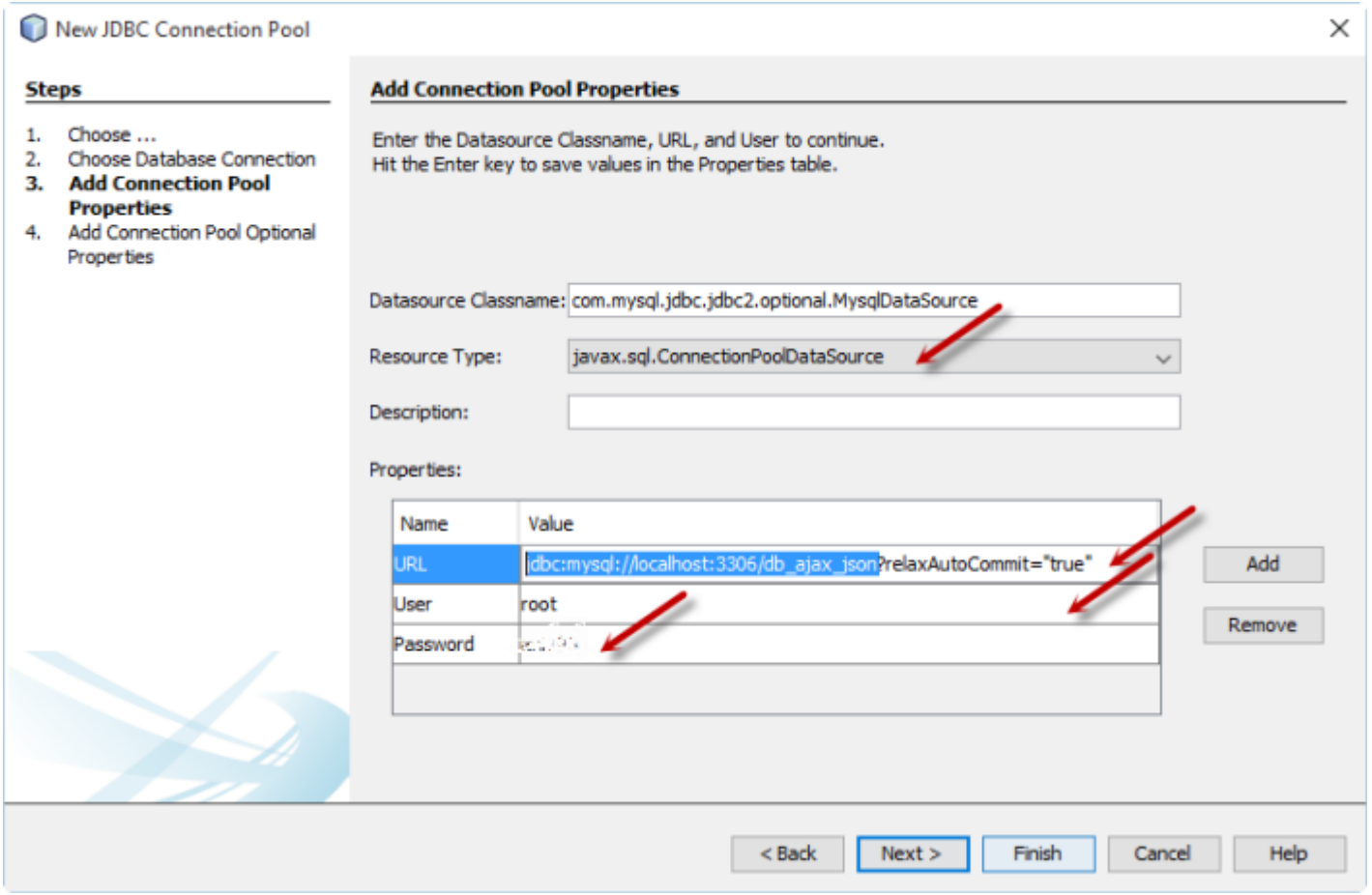




2. En option un pool de connexions peut être ajouté si l'application sera utilisée par plusieurs utilisateurs (application multi-sessions). Le pool de connexions que l'on va appeler cp_ajax_json peut gérer plusieurs connexions à la fois et qui peut être paramétré en termes de nombre de connexions simultannées par exemple et autres paramètres que nous n'évoquons pas ici. Vous allez utiliser la connexion qui vient d'être créée pour le configurer. Pour cela, vous faites un click droit sur le projet puis New --> Other --> GlassFish --> JDBC connection pool comme indiqué ci-dessous :

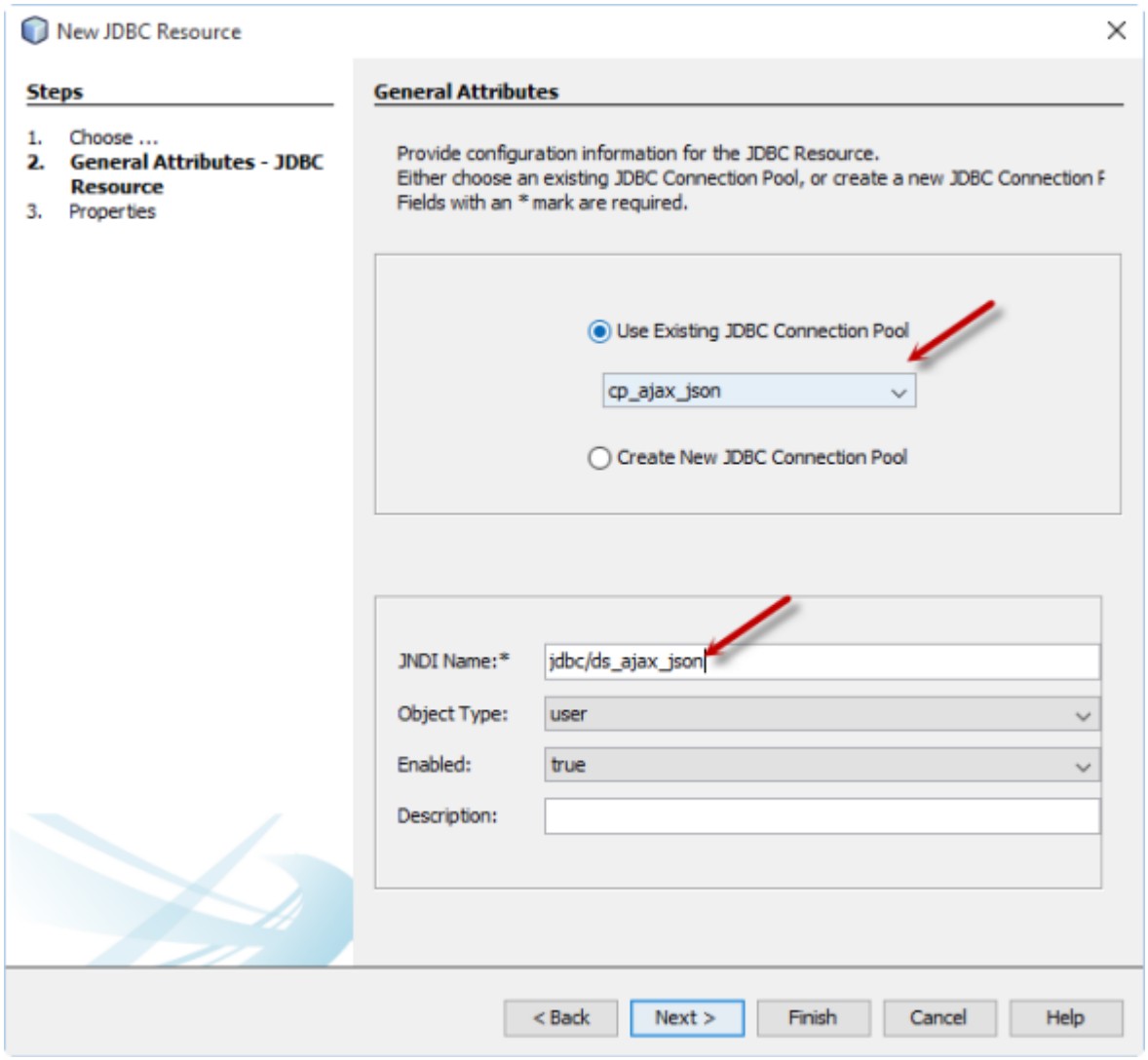

New File
Steps
1. Choose File Type
2. ...
Choose File Type
Project: ajax_json
Filter:
Categories:
Groovy
Hibernate
Web Services
XML
GlassFish
File Types:
JDBC Connection Pool
JDBC Resource
JMS Resource
JavaMail Session
GlassFish Descriptor
Description:
Wizard to create a new JDBC Connection Pool Resource.
< Back
Next >
Finish
Cancel
Help


New JDBC Connection Pool
Steps
1. Choose ...
2. Choose Database Connection
3. Add Connection Pool Properties
4. Add Connection Pool Optional Properties
Choose Database Connection
Provide configuration information for the JDBC Connection Pool.
Either choose an existing database connection to extract information, or ente
Fields with an * mark are required.
JDBC Connection Pool Name:* cp_ajax_json
Extract from Existing Connection:
jdbc:mysql://localhost:3306/db_ajax_json [root on Default schema]
New Configuration using Database:
< Select from the list >
XA (Global Transaction)
< Back
Next >
Finish
Cancel
Help



3. Vous ajouter maintenant la source de données qui va justement utiliser le pool de connexions que l'on vient de créer. Vous procédez comme à l'étape 2, mais vous sélectionnez plutôt dans GlassFish un JDBC Resourse Connexion :
. Le principe de création d'une source de données est le même peu importe le serveur d'applications que

vous utilisez.



Toutes les données saisies aux étapes 1, 2 et 3 sont automatiquement ajoutées par Netbeans dans le fichier de configuration du serveur d'application GlashFich matérialisé par un fichier XML : sun-resources.xml (à noter qu'il arrive qu'il porte un nom différent !) dont voici un aperçu :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE resources PUBLIC "-//Sun Microsystems, Inc.//DTD Application Server
<resources>
  <jdbc-resource enabled="true"
                 jndi-name="jdbc/ds_ajax_json"
                 object-type="user"
                 pool-name="cp_ajax_json">
    <description>MySql JDBC datasource for our project</description>
  </jdbc-resource>
  <jdbc-connection-pool allow-non-component-callers="false" associate-with-th
    <property name="URL" value="jdbc:mysql://localhost:3306/db_ajax_json"/>
    <property name="User" value="root"/>
    <property name="Password" value="      "/>
  </jdbc-connection-pool>
</resources>
```

Par ailleurs, il est à noter que les 3 étapes précédentes peuvent être réalisées directement dans la console WEB de GlashFish ou celle du serveur d'application que vous utilisez.

Conclusion

Dans cet article, nous avons montré comment passer du concept au concret, difficulté à laquelle sont confrontés bon nombre de débutants ou de personnes se reconvertissant au domaine de développement applicatif utilisant notamment les nouvelles technologies du WEB.

Il a été voulu de ne pas aborder le code de l'application proposée afin d'illustrer les différents concepts exposés, car j'estime que la compréhension des concepts et la manière de les matérialiser sont plus importantes que les centaines, voire les milliers ou centaines de milliers de lignes de code à implémenter. Pour construire, il faut d'abord concevoir l'architecture et poser les fondations et ce n'est qu'après ces étapes fondamentales que nos préoccupations se concentrent sur les détails en termes de choix de la stratégie de programmation et du design physique. C'est cela que j'ai tenté de montrer dans cet article que j'espère sera d'utilité pour certains parmi vous qui s'intéressent à ce sujet.

Et la suite

Je vous propose de modifier le projet en téléchargement pour qu'il prend en charge le format XML en plus en tant que choix supplémentaire de l'utilisateur à travers l'unique page de l'application. Je vous propose également d'extérioriser toutes les validations de la servlet UserCRUD dans une classe dédiée et faire appel aux fonctions de validation.

Dans un prochain article, j'espère présenter ce même projet, mais avec l'utilisation d'une autre architeture de vue basée sur l'utlisation de la technologie JSF (Java Server Faces) couplée avec le Framework d'interface graphique (IHM) Prime Faces à mon avis le plus complet du marché en open source dans sa catégorie.