4D Mobile

Wakanda, édité par 4D SAS, est une plate-forme de développement et de publication d'applications Web entièrement basées sur des technologies standard telles que JavaScript et HTML5.

L'architecture "4D Mobile" propose un connecteur afin de mettre en place une liaison directe entre 4D et Wakanda. Avec cette configuration, vous alliez la richesse graphique et fonctionnelle des interfaces Web de dernière génération de Wakanda à la puissance de vos bases de données 4D.

Si vous souhaitez dès maintenant créer votre première liaison entre 4D et Wakanda, vérifiez dans la section **Configuration** que vous disposez de la configuration adéquate et rendez-vous au paragraphe **Exemple au pas à pas**.

Architecture 4D Mobile

Configuration

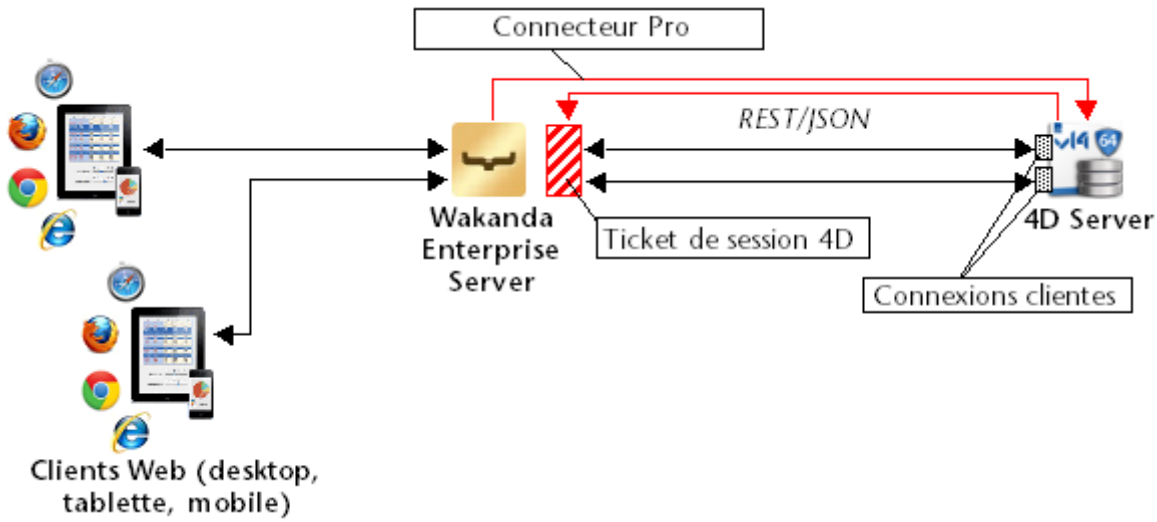
Pour mettre en place une architecture exploitant le connecteur "Pro" 4D / Wakanda, vous avez besoin au minimum de :

- **4D** monoposte (édition *Professional*) pour développer et tester votre solution exploitant le connecteur de 4D Mobile (trois connexions 4D Mobile simultanées sont autorisées dans ce contexte), ou **4D Server** avec un Expansion pack 4D Mobile (deux connexions 4D Mobile autorisées).
- **Wakanda Enterprise Server** ainsi que **Wakanda Enterprise Studio** pour le développement. Ces deux applications peuvent être téléchargées depuis la page de téléchargement de Wakanda (onglet *Enterprise*).
- une base 4D et une application Wakanda devant communiquer entre elles.

Côté 4D, vous devez paramétrer chaque table, attribut et méthode devant être accessible par l'application Wakanda (cf. paragraphe **Configuration de la base 4D**).

Description

L'architecture de 4D Mobile peut être représentée de la manière suivante :



Au démarrage de la solution Wakanda, la liaison est établie par le serveur Wakanda Enterprise avec 4D Server en fonction des paramètres définis dans la boîte de dialogue "Connect to Remote Datastore" ou des méthodes JavaScript de connexion. Une fois la connexion acceptée par 4D Server (cf. paragraphe **Contrôles des accès REST**), un "ticket" de session cliente 4D Mobile est délivré au serveur Wakanda. Ce ticket sera utilisé par le serveur Wakanda pour toutes les requêtes ultérieures.

Via cette liaison, le serveur Wakanda peut potentiellement accéder à deux types de ressources de la base 4D :

- les tables et leurs attributs (et leurs données)
- les méthodes projet

Lorsqu’elles sont autorisées, ces ressources sont utilisables directement côté Wakanda, comme si elles appartenaient au catalogue local de l’application (leur accès est transparent depuis l’application Wakanda).

Lorsqu’un client Web envoie au serveur Wakanda une requête nécessitant un accès à la base 4D, la requête est acheminée au serveur 4D en utilisant le ticket courant et une connexion 4D Mobile est ouverte sur le poste 4D Server. La connexion restera ouverte tant que l’utilisateur effectuera des requêtes et sera refermée par défaut au bout de 60 minutes d’inactivité (timeout). Ce délai par défaut peut être modifié dans les paramètres initiaux de connexion. Si au cours de la session, le nombre de licences correspondant au nombre de connexions 4D Mobile autorisées sur 4D Server est atteint, un message d’erreur est retourné au serveur Wakanda.

Exemple au pas à pas

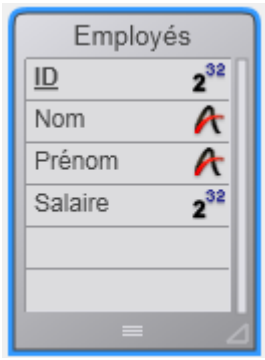
Cet exemple vous propose de découvrir rapidement les fonctionnalités 4D Mobile au travers d'un exemple construit au pas à pas. Dans cet exemple, nous allons :

- créer et configurer une base 4D
- créer une application Wakanda contenant une seule page
- afficher les données de la base 4D dans la page Wakanda.

Pour simplifier l'exemple, nous allons utiliser une application 4D et une application Wakanda exécutées sur le même poste. Bien entendu, vous pourrez utiliser une architecture distante.

1 - Création et configuration de la base 4D

1. Lancez votre application 4D ou 4D Server et créez une nouvelle base. Nommez-la par exemple "Emp4D".
2. Dans l'éditeur de Structure, créez la table [Employés] et ajoutez-lui les champs suivants :
   - Nom (texte)
   - Prénom (texte)
   - Salaire (entier long)

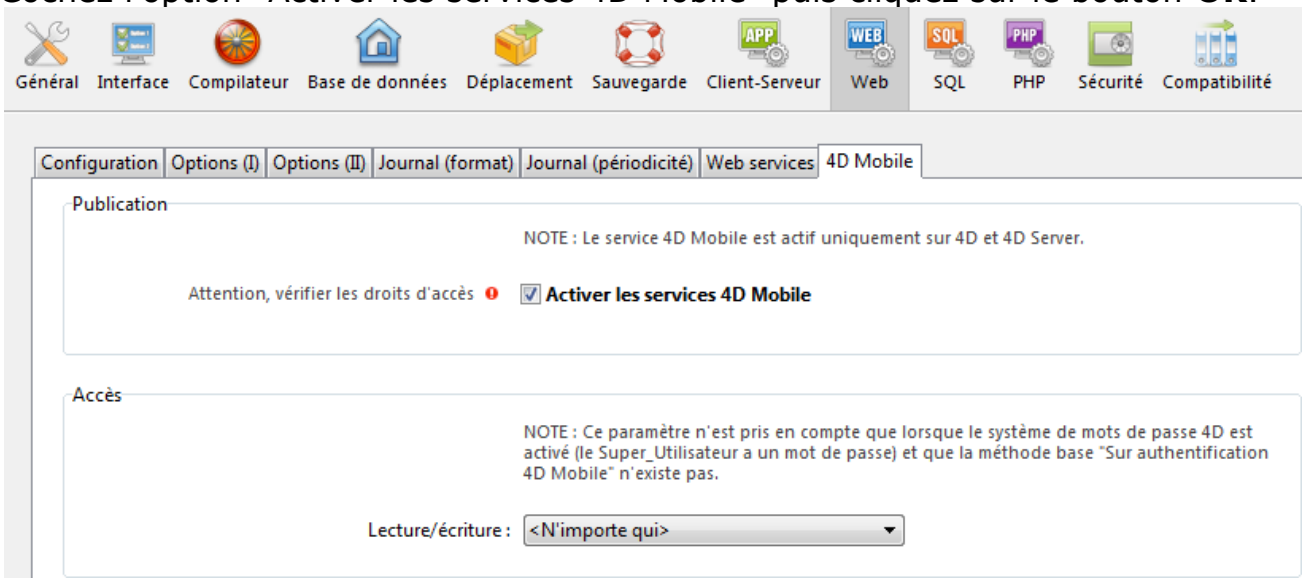
   L'attribut "Exposer avec le service 4D Mobile" est sélectionné par défaut pour la table et chaque champ, conservez ce paramétrage.
3. Cliquez sur le bouton **Tables**, laissez 4D créer les formulaires par défaut et créez quelques employés :

   Emp4D - Employés : 3 enregistrement(s) sur 3

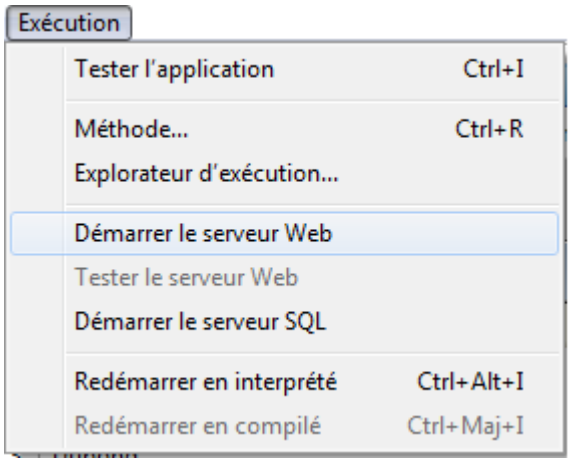
ID :	Nom :	Prénom :	Salaire :
1	Martin	Jean	1800
2	Durand	Pierre	2400
3	Dupond	Marie	1950

4. Affichez la boîte de dialogue des Propriétés de la base, page **Web**, onglet **4D Mobile**.

5. Cochez l’option "Activer les services 4D Mobile" puis cliquez sur le bouton **OK**.



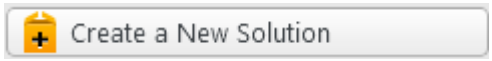
6. Dans le menu **Exécution**, choisissez la commande **Démarrer le serveur Web** :



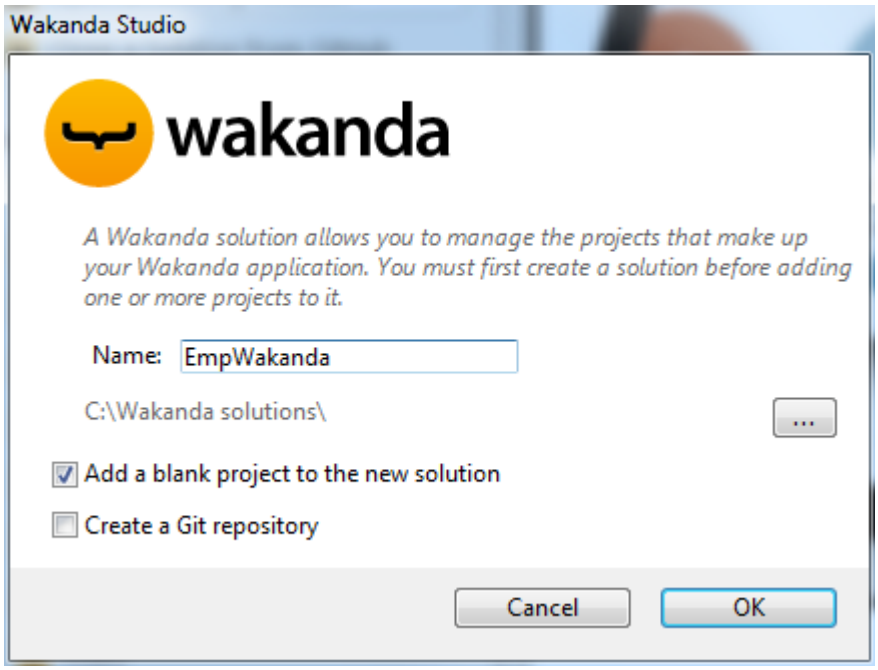
La base 4D est dès lors prête à répondre aux requêtes 4D Mobile de Wakanda. A noter que pour simplifier cet exemple, nous ne contrôlons pas les accès 4D Mobile. Dans un contexte de production ou d’architecture ouverte, il est indispensable de sécuriser les accès 4D Mobile via REST (cf. paragraphe **A propos de la sécurité des applications 4D Mobile**).

2 - Créer l’application Wakanda

1. Lancez l’application "Wakanda Enterprise Studio" et cliquez sur le bouton **Create a New Solution** :

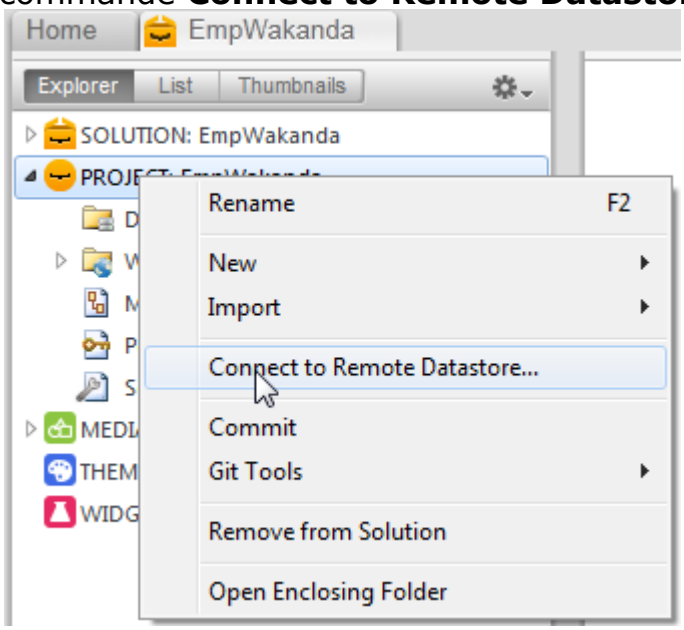


2. Dans la boîte de dialogue de création, saisissez par exemple "EmpWakanda" et cliquez sur **OK**:

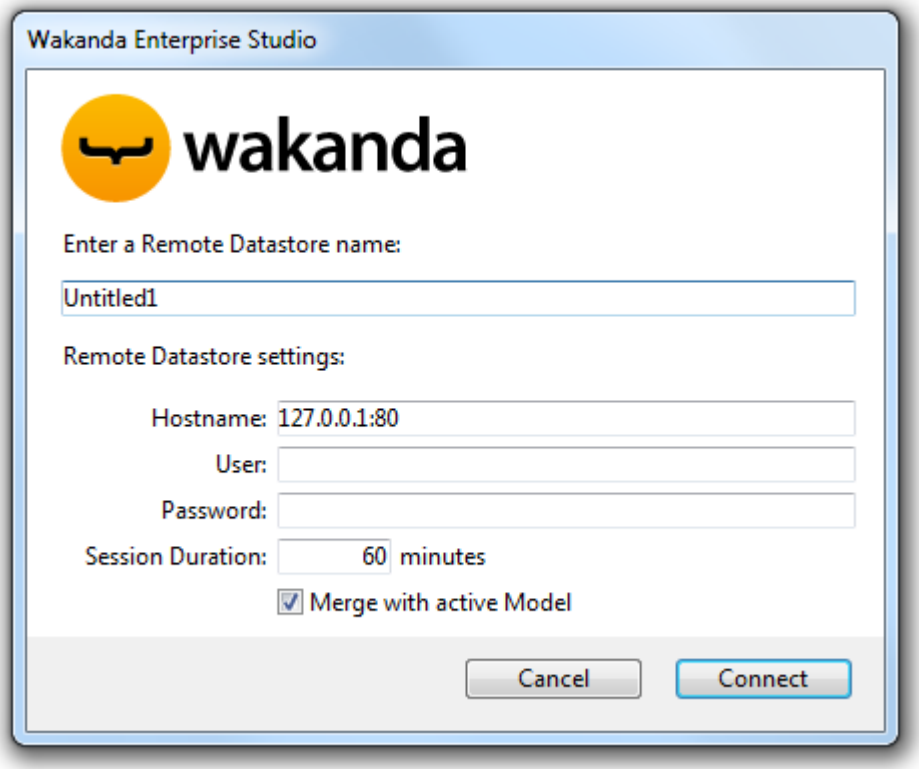


Le projet d'application est créé, les éléments par défaut apparaissent dans l'Explorer de Wakanda Studio, dans la partie gauche de la fenêtre.

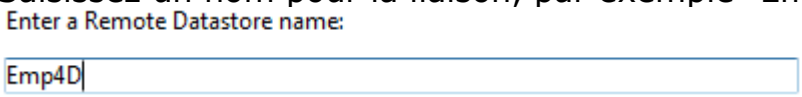
3. Cliquez avec le bouton droit de la souris sur la ligne PROJECT et sélectionnez la commande **Connect to Remote Datastore...** dans le menu contextuel.



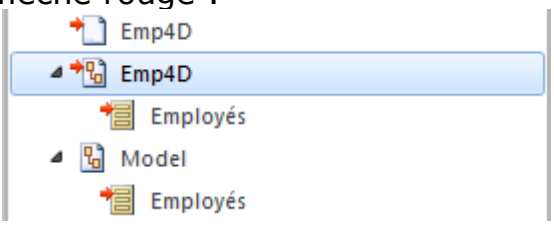
La boîte de dialogue de connexion apparaît :



4. Saisissez un nom pour la liaison, par exemple "Emp4D" :



Il s'agit du nom local de la liaison, telle qu'elle apparaîtra dans Wakanda Enterprise Studio. Vous pouvez saisir n'importe quel nom, toutefois pour simplifier nous utilisons le nom de la base 4D.
5. (optionnel) Si votre 4D Server est situé sur une autre machine que Wakanda Enterprise Studio, saisissez son hostname ou son adresse IP comme paramètre Hostname. Sinon, vous pouvez conserver l'adresse locale "127.0.0.1:80" (ou "localhost").
6. Conservez les autres paramètres par défaut et cliquez sur le bouton **Connect**. Au bout de quelques instants, vous pouvez constater que le modèle externe "Emp4D" apparaît bien parmi les fichiers de l'application Wakanda et que la table [Employés] de l'application 4D est également listée dans la liste des datastore class du modèle local. Les éléments externes sont signalés par une flèche rouge :



Note : Le premier fichier Emp4D contient les paramètres de connexion.

En cas de problème...

Si à ce stade, la table n'apparaît pas dans la liste, vérifiez que :

- aucun service ou logiciel tiers (messagerie instantanée par exemple) n'est en conflit avec le port de publication (80 par défaut) du serveur HTTP de 4D,
- côté 4D, le serveur Web de 4D est bien lancé, les services 4D Mobile démarrés et la table exposée,
- l'adresse passée dans le paramètre "Hostname" est valide.

Pour vérifier que le serveur de 4D répond bien aux requêtes REST, vous pouvez saisir les URLs suivants dans votre navigateur :

```
<adresse>/rest/$catalog/$all
```

(retourne les tables exposées avec le service 4D Mobile)

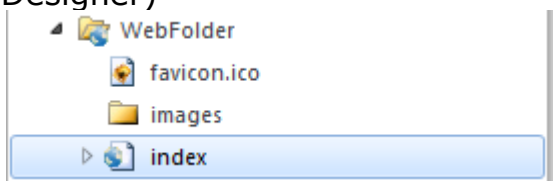
```
<adresse>/rest/ma_table/ma_methode
```

(retourne le résultat de la méthode - si elle retourne un résultat)

3 - Afficher les données 4D via un widget Wakanda

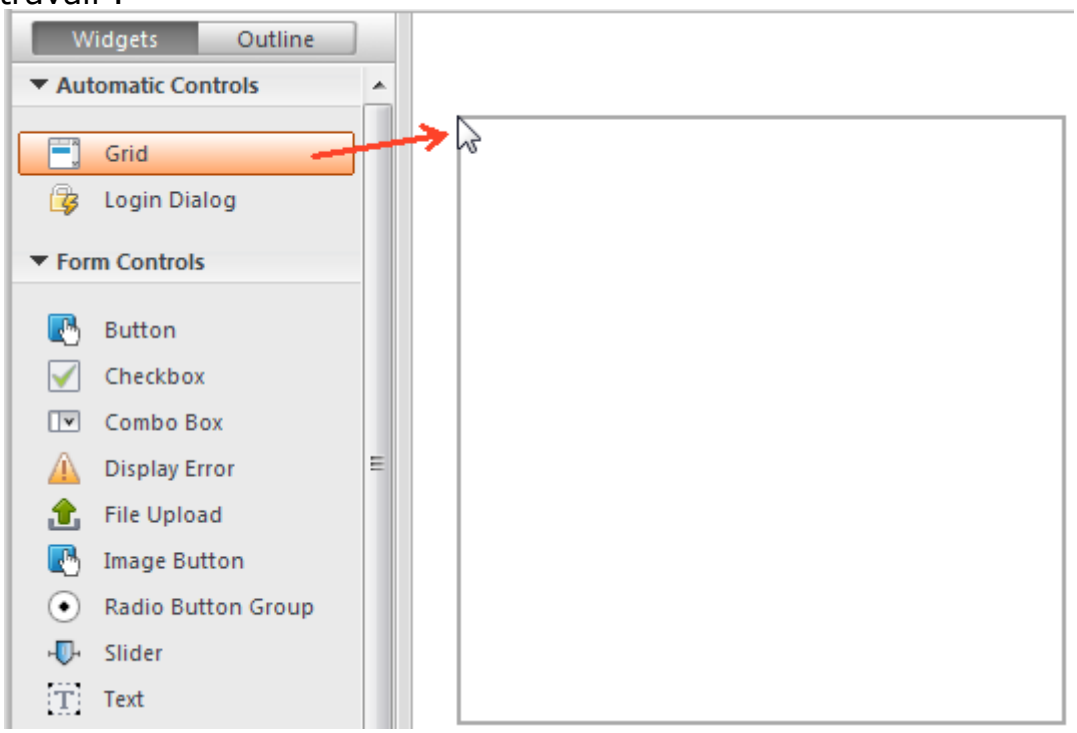
Nous allons maintenant associer la table 4D à un widget Wakanda par simple glisser-déposer, lancer Wakanda Server Enterprise et visualiser les données.

1. Ouvrez le dossier "WebFolder" dans l'Explorer et double-cliquez sur la page Index afin d'ouvrir le Concepteur d'interfaces graphiques de Wakanda (GUI Designer)

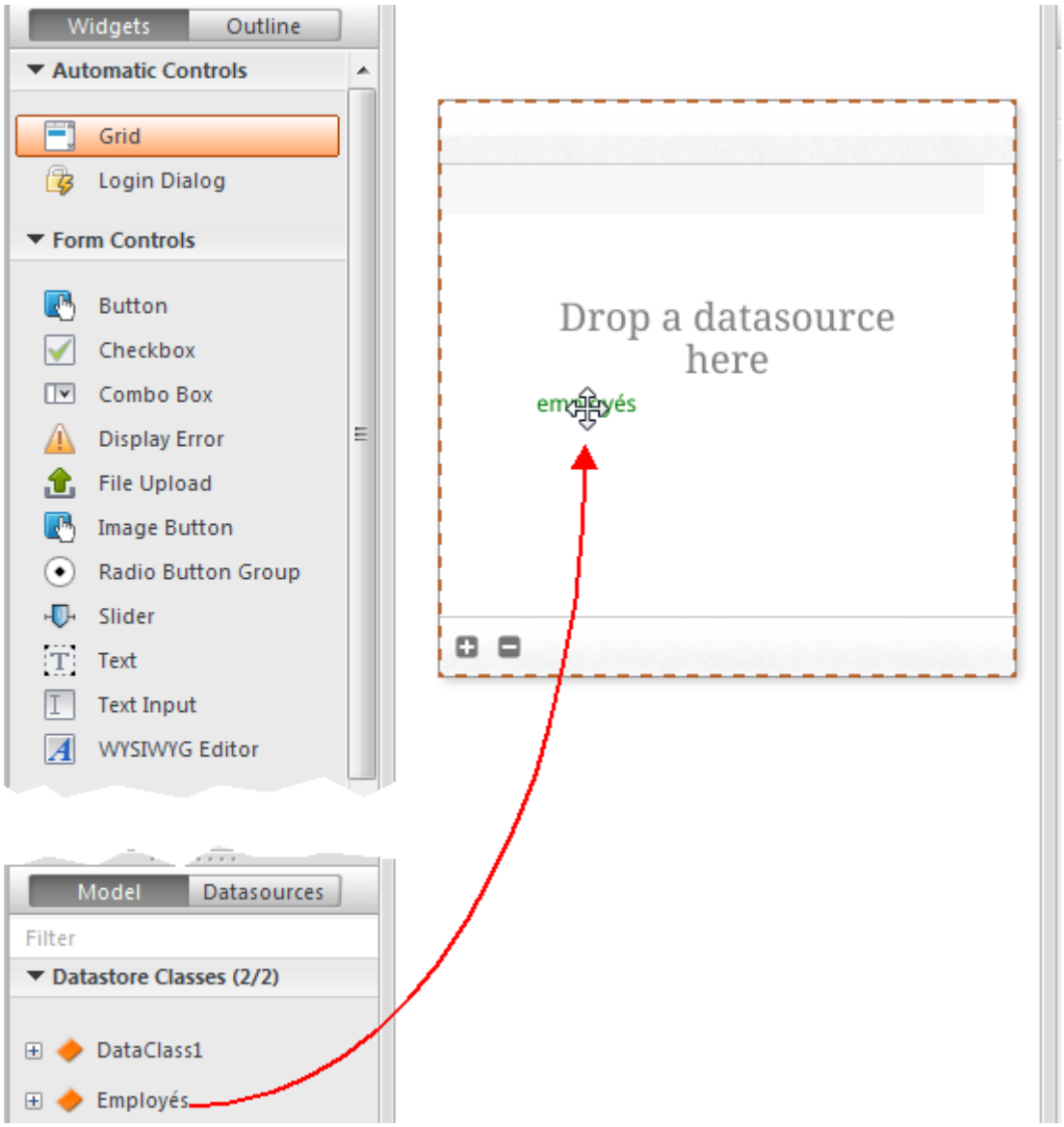


Note : Le dossier "WebFolder" contient les éléments destinés à la publication Web de votre projet. "Index" est la page par défaut du projet.

2. Dans la liste des Widgets, cliquez sur "Grid" et déposez-le dans la zone de travail :



3. Dans la liste des Datastore Classes du modèle, cliquez sur "Employés" et déposez-le sur la grid que vous venez de créer :

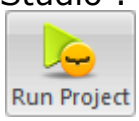


A cet instant, l'éditeur crée automatiquement pour vous une *datasource* basée sur la classe "Employés", qui sera chargée de gérer le contenu du widget. Cette *datasource* est un objet JavaScript géré par Wakanda, nommé par défaut "employés", c'est-à-dire le nom de la classe avec la première lettre en minuscule.
Le widget affiche une prévisualisation de son contenu. Vous pouvez l'agrandir afin d'afficher la totalité des champs de la *datasource* :

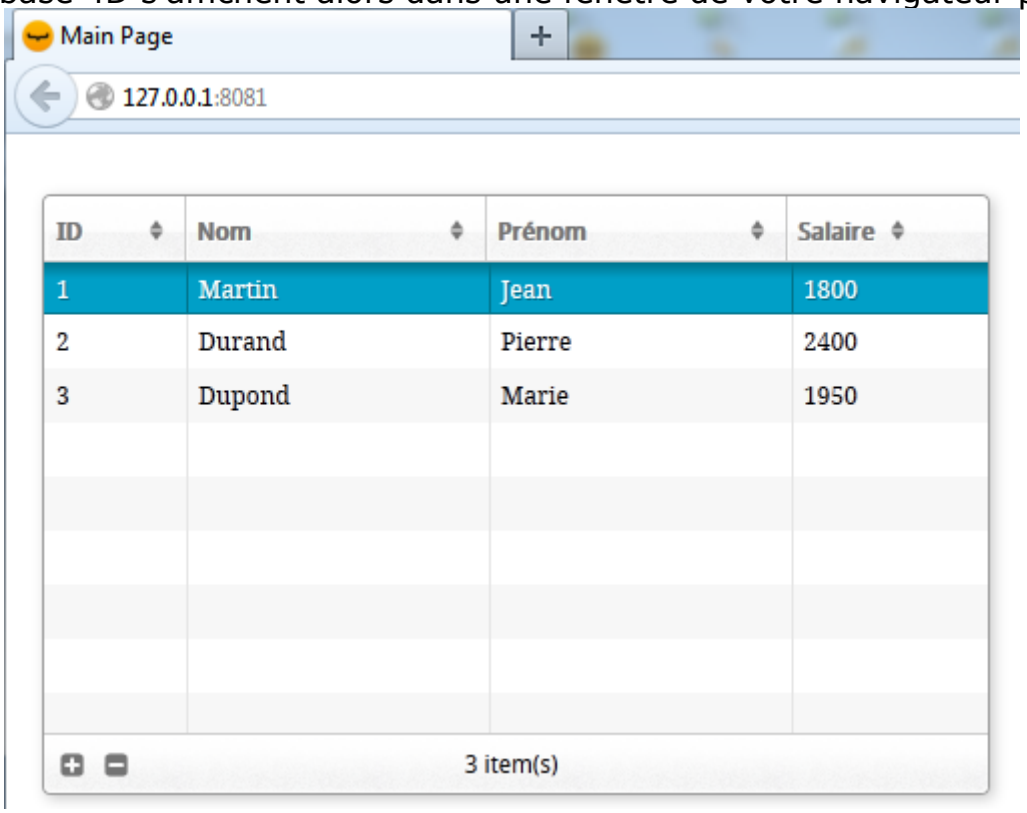
L'association entre la *datasource* et le widget est alors établie.

4. Cliquez sur le bouton **Save**  dans la barre d'outils de l'éditeur.

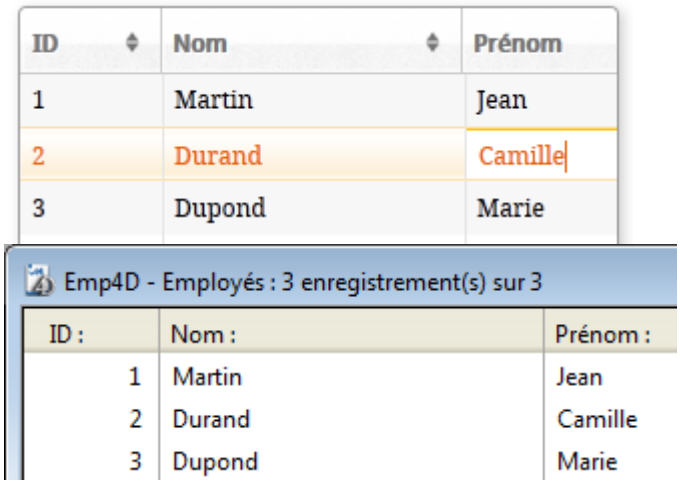
   Nous allons maintenant visualiser les données via un navigateur.
5. Cliquez sur le bouton **Run projet** dans la barre d'outils de Wakanda Enterprise Studio :

   Cette action démarre Wakanda Enterprise Server et publie l'application "EmpWakanda". Grâce à la liaison 4D Mobile mise en place, les données de la base 4D s'affichent alors dans une fenêtre de votre navigateur par défaut :



   Vous pouvez tester les propriétés dynamiques de la liaison en modifiant les données côté Web. Ici par exemple, le prénom "Pierre" est changé en "Camille", 4D reflète immédiatement la modification :



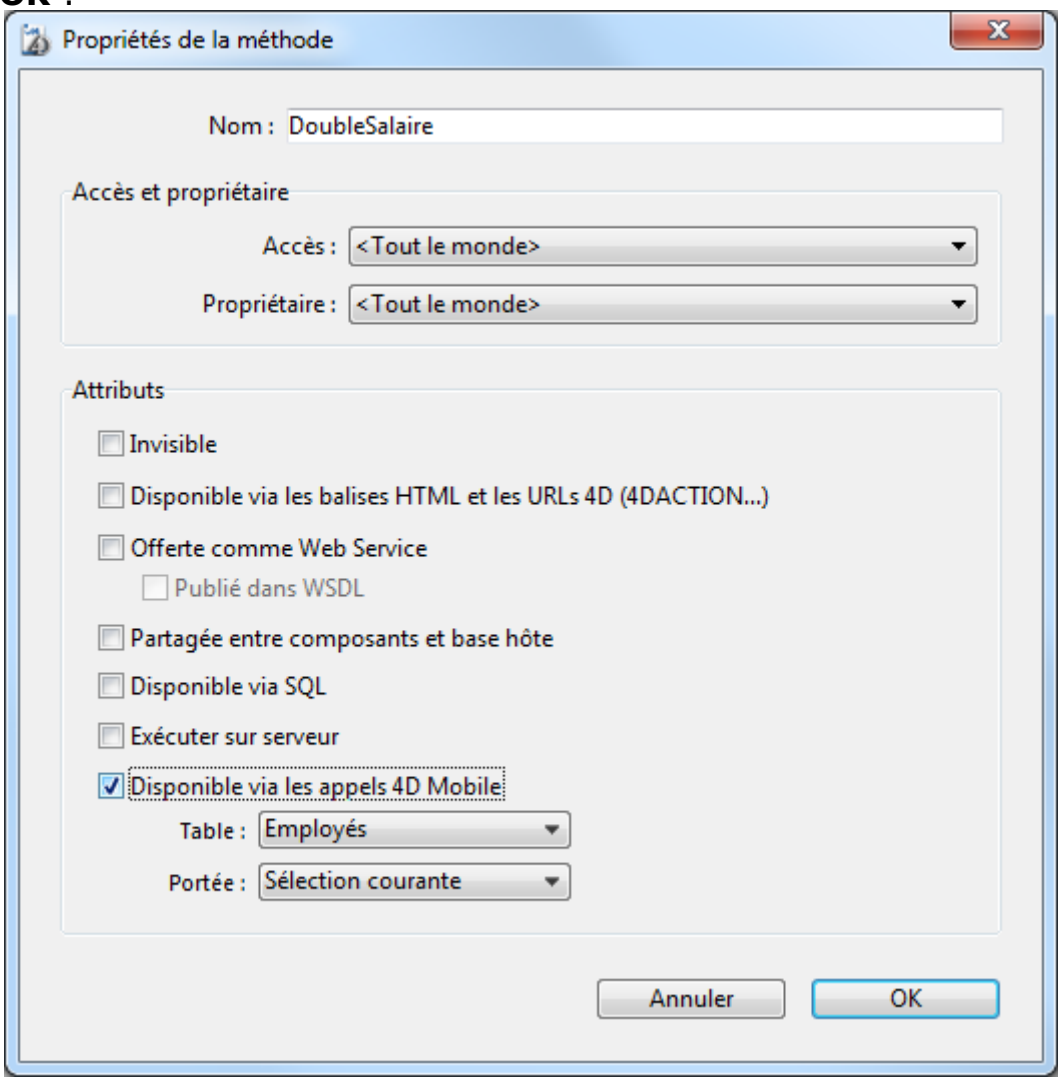
4 - Créer et appeler une méthode 4D

Nous allons maintenant créer une méthode projet très simple côté 4D et l'exécuter depuis notre page Web. Cette méthode doublera tous les salaires.

1. Côté 4D, créez une méthode projet nommée ***DoubleSalaire*** et saisissez le code suivant :

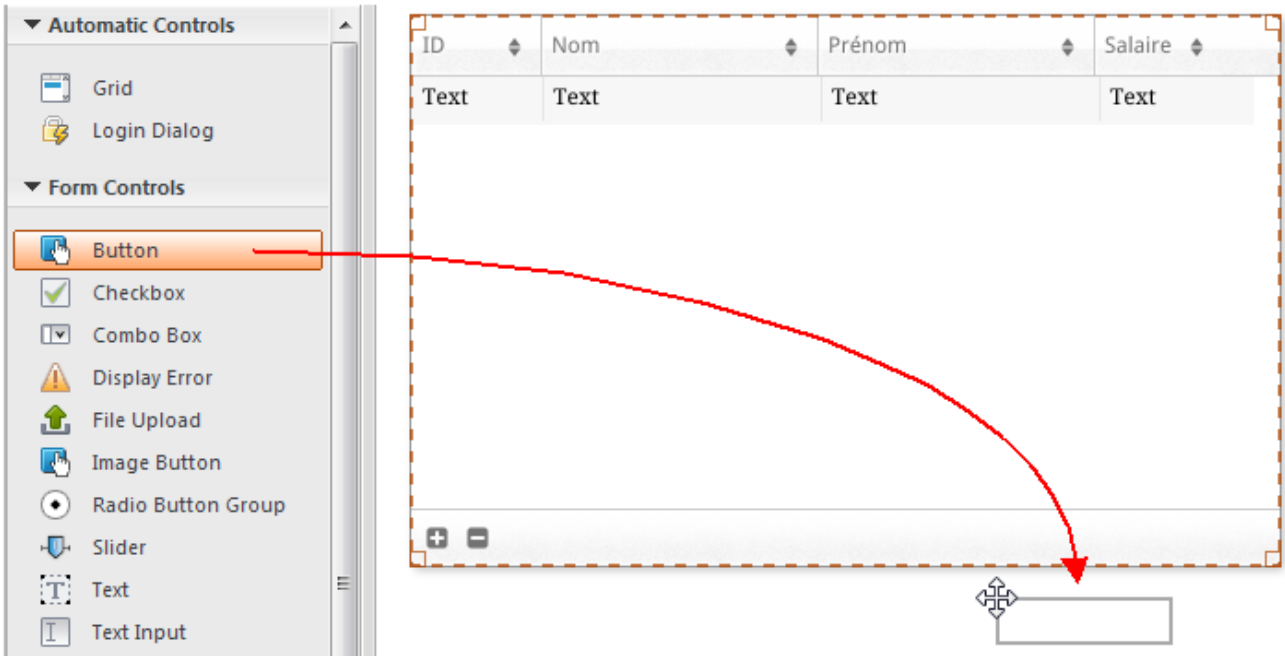
```
DEBUT SELECTION([Employés])
Tant que(Non(Fin de selection([Employés])))
    [Employés]salaire:=[Employés]salaire*2
    STOCKER ENREGISTREMENT([Employés])
    ENREGISTREMENT SUIVANT([Employés])
Fin tant que
```

2. Paramétrez les propriétés des appels 4D Mobile de la méthode et cliquez sur **OK** :

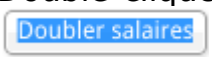


Dans Wakanda, les méthodes de classe s'appliquent à l'un des contextes suivants : l'*entity* (enregistrement), l'*entity collection* (sélection) ou la *datastore class* (tous les enregistrements). Il est nécessaire de préciser ce contexte côté 4D.

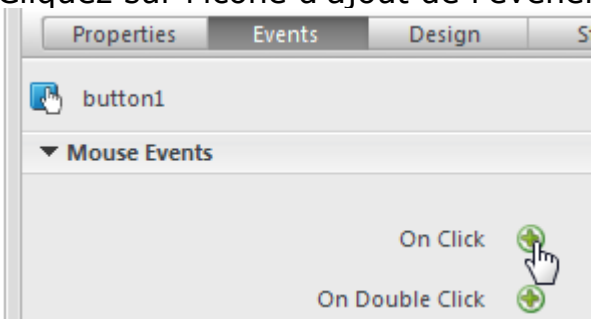
3. Côté Wakanda Enterprise Studio, retournez à la page **Index** dans le *GUI Designer* et ajoutez un bouton depuis la liste des widgets :



4. Double-cliquez sur le bouton et nommez-le, par exemple, "Doubler salaires" :



5. Assurez-vous que le bouton "Doubler salaires" est bien sélectionné et cliquez sur le bouton **Events** dans la zone de droite du *GUI Designer*.
6. Cliquez sur l'icône d'ajout de l'événement "On Click" :



L'éditeur de code s'affiche, vous permettant de saisir le code à exécuter en cas de clic sur le bouton. Nous allons simplement appeler la méthode ***DoubleSalaire*** de 4D puis, dans la fonction de callback (*onSuccess*), provoquer le rechargement de tous les enregistrements.

7. Saisissez le code suivant :

```
sources.employés.DoubleSalaire({ onSuccess:function(event){
sources.employés.allEntities(); }});
```

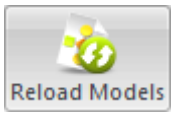
Dans l'éditeur de code :

```
button1.click = function button1_click (event)
{
    sources.employés.DoubleSalaire({
            onSuccess:function(event){
                sources.employés.allEntities();
            }});
};
```

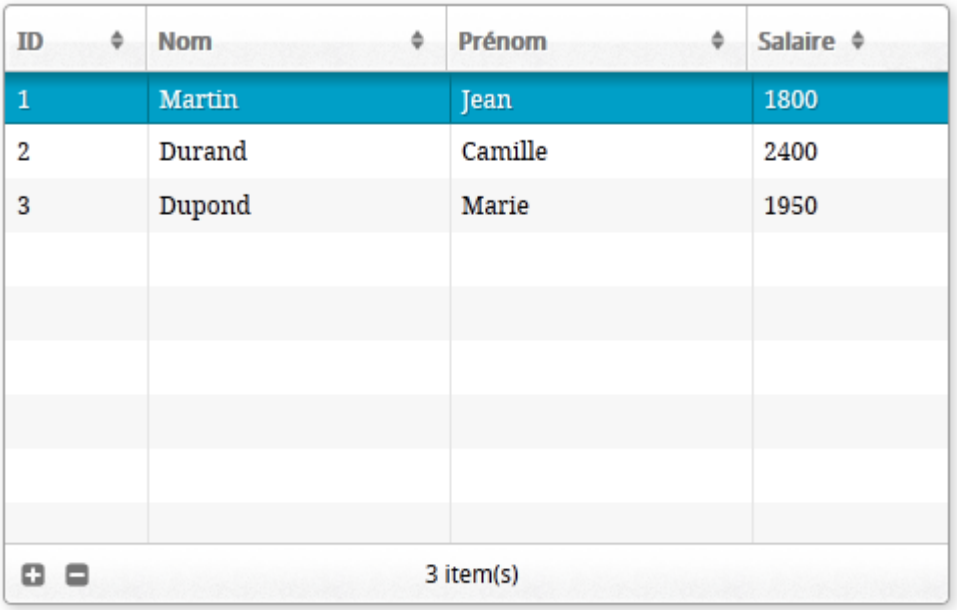
Notez bien le "e" minuscule de "employés" : nous utilisons la *datasource* créée automatiquement lors de l'association de la classe et du widget.

8. Cliquez sur le bouton **Save**  dans la barre d'outils de l'éditeur.

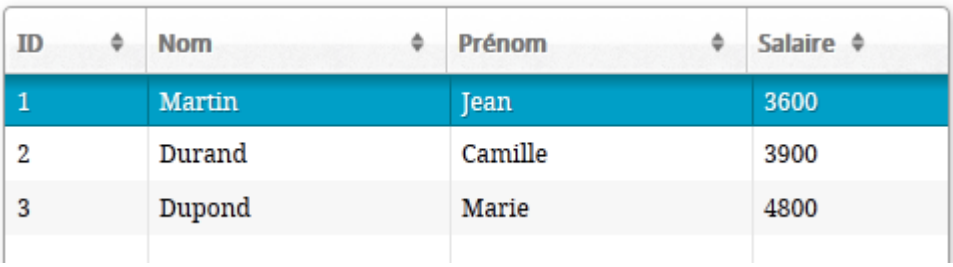
Nous allons pouvoir tester l'appel de la méthode 4D. Auparavant, vous devez recharger le modèle sur Wakanda Enterprise Server.

9. Cliquez sur le bouton **Reload Models** 

dans la barre d'outils de Wakanda Enterprise Studio.
10. Rafraîchissez la page de votre navigateur afin de faire apparaître le bouton **Doubler salaires** et cliquez sur ce bouton :



Vous constatez que les valeurs des salaires ont doublé :



A noter que cet exemple est uniquement destiné à montrer les principes de mise en place du connecteur 4D / Wakanda, les méthodes simples proposées ne sont pas utilisables dans un contexte de production.

Configuration de la base 4D

Pour des raisons de sécurité et de performance, l'accès aux tables, données et méthodes de la base 4D via des requêtes 4D Mobile (serveurs Wakanda) doit être activé et explicitement autorisé. Vous devez configurer trois niveaux d'accès :

- démarrage des services 4D Mobile,
- contrôle des accès 4D Mobile (optionnel mais recommandé),
- définition individuelle de l'exposition de chaque objet de la base (table, attribut ou méthode projet) aux services 4D Mobile en fonction de vos besoins. Par défaut :
  - toutes les tables et tous les attributs sont accessibles à 4D Mobile,
  - les méthodes projet ne sont pas accessibles à 4D Mobile.

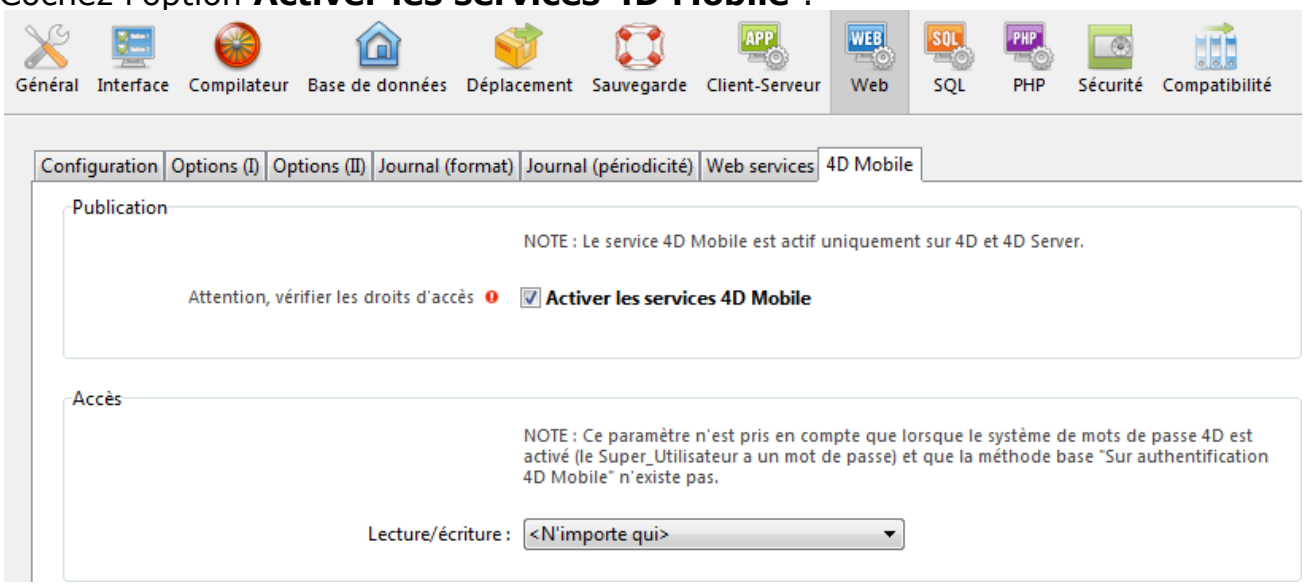
Activer les services 4D Mobile

Par défaut, 4D Server ne répond pas aux requêtes 4D Mobile. Vous devez démarrer les services 4D Mobile afin que ces requêtes soient traitées et que le connecteur 4D / Wakanda puisse être mis en place.

Note : Les services 4D Mobile utilisent le serveur HTTP de 4D. Par conséquent, assurez-vous que le serveur Web de 4D ou 4D Server est démarré.

Pour activer les services 4D Mobile :

1. Dans les Propriétés de la base, affichez la page Web/4D Mobile.
2. Cochez l'option **Activer les services 4D Mobile** :



Le message d'alerte "Attention, vérifier les droits d'accès" s'affiche afin d'attirer votre attention sur le fait que lorsque les services 4D Mobile sont activés, par défaut l'accès aux objets de la base est libre tant que les accès 4D Mobile (via REST) n'ont pas été contrôlés (cf. ci-dessous).

Contrôles des accès 4D Mobile

Le contrôle des accès 4D Mobile permet d'autoriser ou non l'ouverture d'une session côté 4D à la suite d'une requête Wakanda.

Dans le cadre d'un accès 4D Mobile, les identifiants contrôlés sont le nom et le mot de passe envoyés lors de la demande de connexion effectuée par :

- la boîte de dialogue "Connect to Remote Datastore" de Wakanda Enterprise Studio
- les méthodes SSJS **mergeOutsideCatalog()**, **openRemoteStore()** ou **addRemoteStore()**.

Au niveau global, les accès 4D Mobile peuvent être contrôlés de deux manières :

- soit automatiquement, via les mots de passe 4D,
- soit par programmation via la **Méthode base Sur authentification 4D Mobile**.

Ces deux modes de contrôle sont exclusifs : si une **Méthode base Sur authentification 4D Mobile** est définie, le contrôle des accès automatique par mots de passe 4D est désactivé.

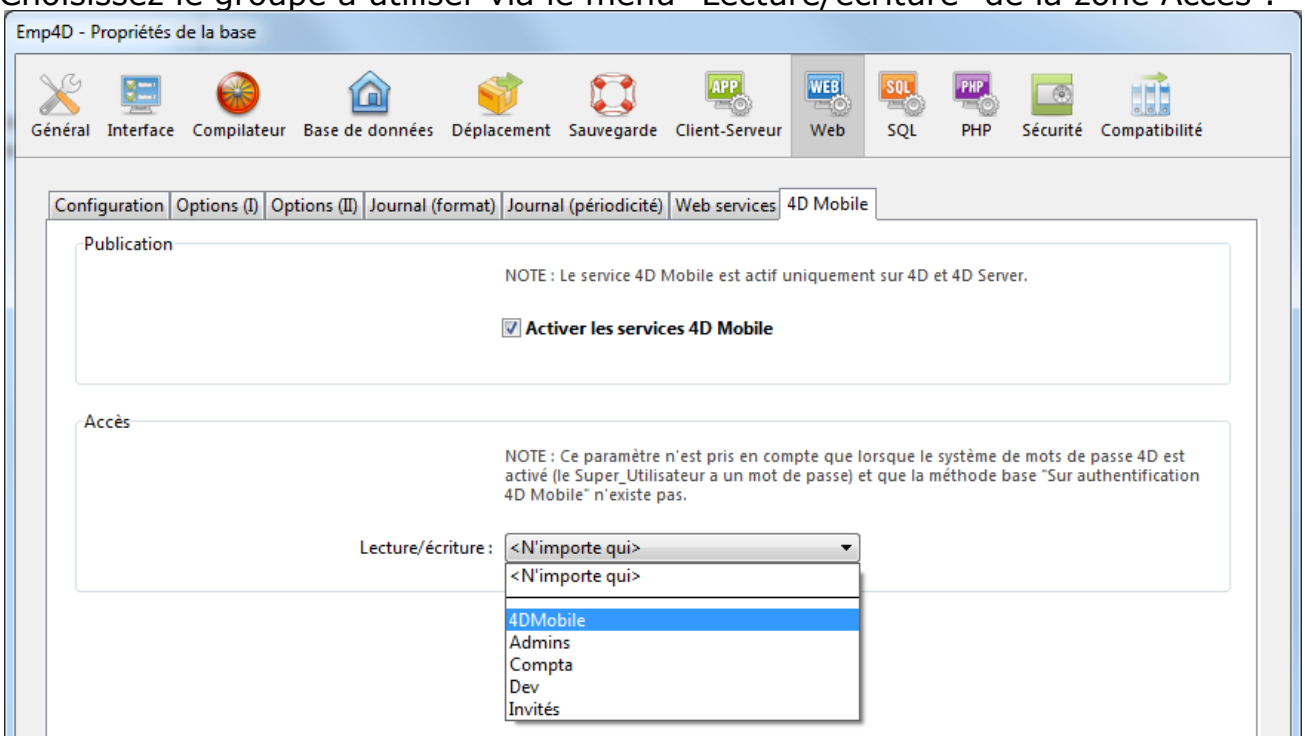
Attention, si aucun de ces deux modes de contrôle n'est activé, les accès 4D Mobile à la base sont toujours acceptés (déconseillé).

Contrôles automatiques par mot de passe 4D

4D vous permet de désigner le groupe d'utilisateurs qui sera autorisé à établir la connexion au serveur 4D depuis l'application Wakanda.

Pour désigner le compte d'ouverture de session :

1. Dans les Propriétés de la base, affichez la page Web/4D Mobile.
2. Choisissez le groupe à utiliser via le menu "Lecture/écriture" de la zone Accès :



Par défaut, le menu affiche **<N'importe qui>**, ce qui signifie que les accès 4D Mobile sont ouverts à tous les utilisateurs.

Une fois que vous avez désigné un groupe, seul un compte d'utilisateur 4D appartenant à ce groupe pourra être utilisé pour accéder à 4D via une requête Wakanda -- pour ouvrir une session sur le serveur 4D via la méthode **mergeOustideCatalog()** par exemple. Si un compte n'appartenant pas à ce groupe est utilisé, 4D retourne une erreur d'authentification à l'expéditeur de la requête.

A noter que pour que ce paramétrage soit effectif :

- le système de mots de passe de 4D doit être activé (un mot de passe doit avoir été attribué au Super_Utilisateur),
- la **Méthode base Sur authentification 4D Mobile** ne doit pas être définie. Si elle existe, 4D ne tient pas compte des paramétrages d'accès définis dans les Propriétés de la base.

Utiliser la Méthode base Sur authentification 4D Mobile

La **Méthode base Sur authentification 4D Mobile**vous permet de contrôler de manière personnalisée l'ouverture des sessions 4D Mobile sur 4D. Lorsqu'elle est définie, elle est automatiquement appelée par 4D ou 4D Server lorsqu'une requête 4D Mobile est reçue par le serveur HTTP.

Lorsque la demande d'ouverture de session 4D Mobile provient de Wakanda Server (cas général), les identifiants de connexion sont fournis dans l'en-tête de la requête. La méthode base **Sur authentification 4D Mobile** est appelée afin de vous permettre d'évaluer ces identifiants. Vous pouvez utiliser la liste des utilisateurs de la base 4D ou votre propre table d'identifiants.

Pour plus d'informations, reportez-vous à la description de la **Méthode base Sur authentification 4D Mobile** dans le manuel *Langage* de 4D.

Définir les objets 4D exposés en 4D Mobile

Une fois que les services 4D Mobile sont activés dans la base 4D, par défaut une session 4D Mobile peut accéder à toutes les tables et tous les champs de la base, et donc utiliser leurs données. Par exemple, si votre base contient une table [Employee], il est possible d'écrire côté Wakanda Server :

```
var emp=ds.Employee.query("name == 'Martin'");      //Retourner tous les employés dont le champ nom vaut 'Martin'
```

Note : Les tables et/ou champs 4D ayant l'attribut "Invisible" sont également exposés en 4D Mobile par défaut.

Le serveur Wakanda peut également accéder aux méthodes projet de la base 4D. Toutefois, par défaut, cet accès est inactivé pour des raisons de sécurité.

Si vous souhaitez personnaliser la liste des objets de votre base accessibles en 4D Mobile, vous devez :

- désactiver l'exposition de chaque table et/ou champ que vous souhaitez masquer,
- activer l'exposition de chaque méthode projet à laquelle vous souhaitez donner accès.

Lorsqu'une requête 4D Mobile tente d'accéder à une ressource (table ou méthode projet) non autorisée, 4D retourne une erreur.

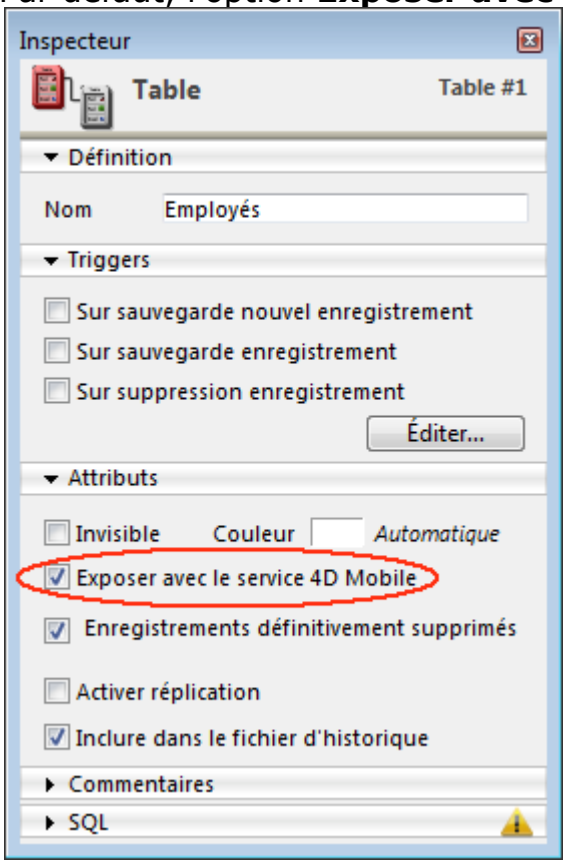
Exposition des tables

Par défaut, toutes les tables sont exposées en 4D Mobile.

Pour des raisons de sécurité, vous pouvez souhaiter ne pas exposer certaines tables de votre base aux appels 4D Mobile. Par exemple, si vous avez créé une table [Users] dans laquelle vous stockez les noms et mots de passe des utilisateurs, il est préférable de ne pas l'exposer.

Pour modifier l'exposition 4D Mobile d'une table :

1. Affichez l'Inspecteur de table dans l'éditeur de Structure et sélectionnez la table à modifier.
   Par défaut, l'option **Exposer avec le service 4D Mobile** est cochée :



2. Désélectionnez l'option **Exposer avec le service 4D Mobile**.
   OU
   Effectuez l'opération inverse pour exposer une table désélectionnée.
   Répétez l'opération pour chaque table dont vous souhaitez modifier l'exposition.

Exposition des champs

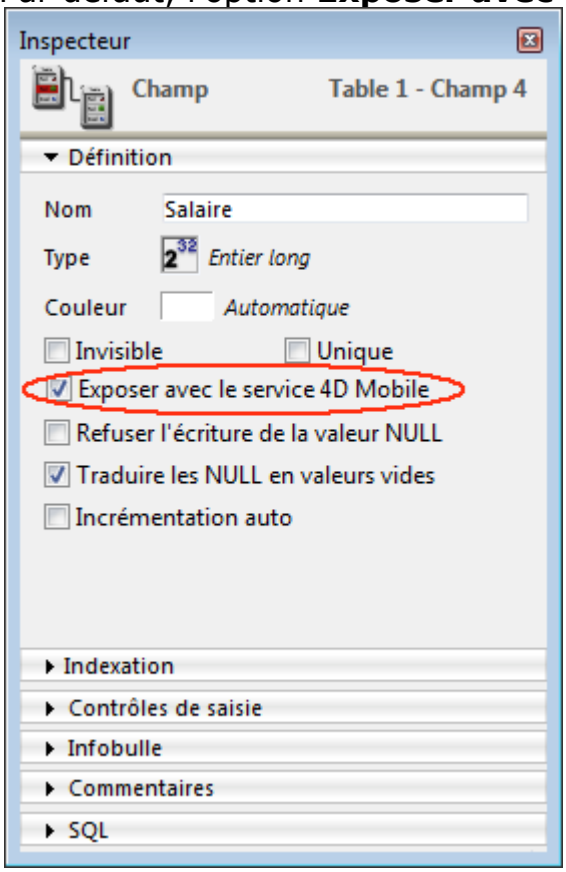
Par défaut, tous les champs de la base 4D sont exposés en 4D Mobile.

Vous pouvez souhaiter ne pas exposer à 4D Mobile certains champs de vos tables. Par exemple, vous pouvez vouloir ne pas exposer le champ [Employés]Salaire.

Pour modifier l'exposition 4D Mobile d'un champ :

1. Affichez l'Inspecteur de champ dans l'éditeur de Structure et sélectionnez le champ à modifier.

Par défaut, l'option **Exposer avec le service 4D Mobile** est cochée :



2. Désélectionnez l'option **Exposer avec le service 4D Mobile** pour le champ.
   OU
   Effectuez l'opération inverse pour exposer un champ désélectionné.
   Répétez l'opération pour chaque champ que vous souhaitez modifier.

A noter que pour qu'un champ soit accessible via 4D Mobile, la table parente doit également l'être. Si la table parente n'est pas exposée, aucun champ de la table ne sera accessible, quel que soit son statut. Grâce à ce mécanisme, vous pouvez activer ou désactiver temporairement l'exposition 4D Mobile d'une table, la valeur individuelle de l'option d'exposition de chaque champ n'est pas modifiée.

Exposition des méthodes projet

Par défaut, aucune méthode projet n'est exposée en 4D Mobile.

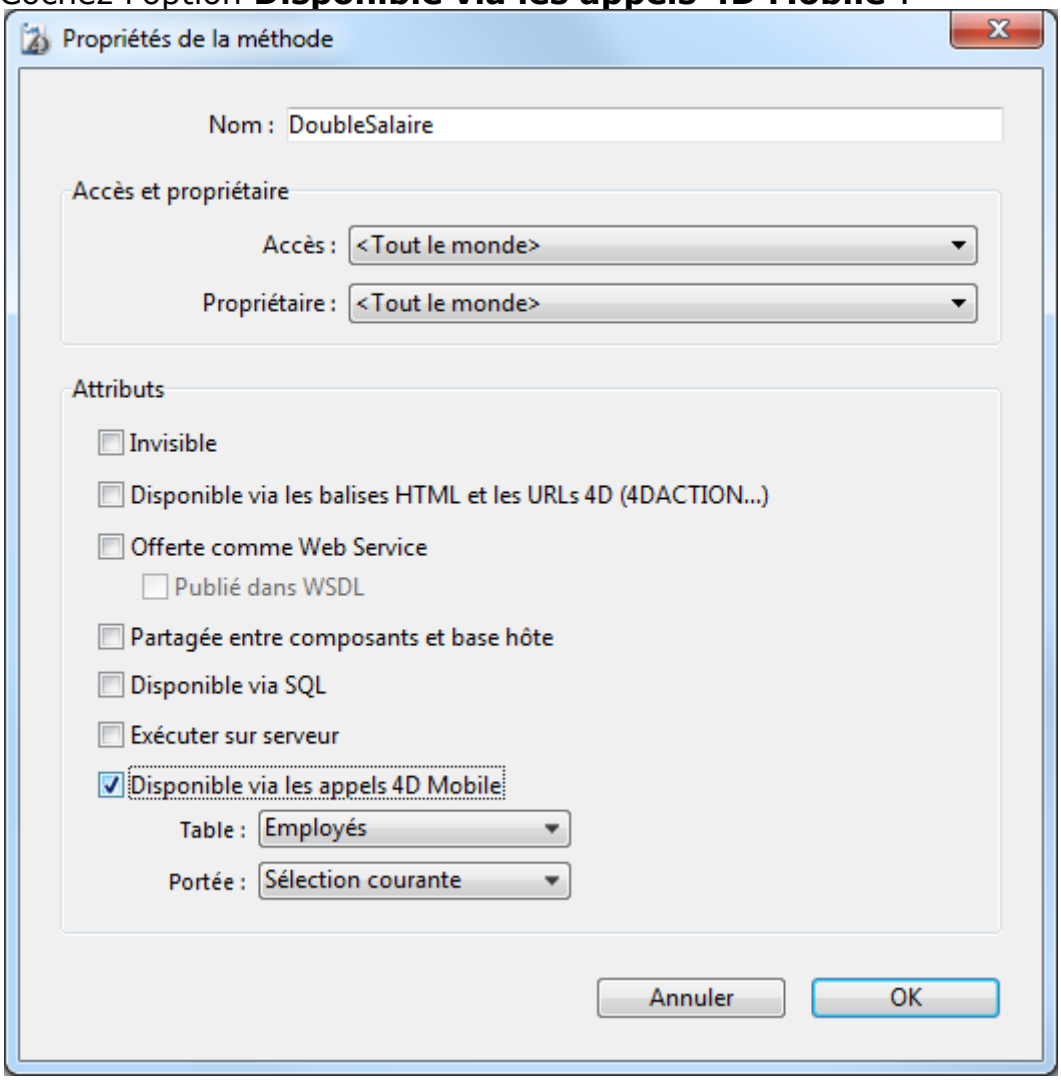
Vous pouvez souhaiter rendre accessibles via 4D Mobile une ou plusieurs méthode(s) projet de votre base 4D. Pour cela, vous devez cocher l'option appropriée et définir le contexte d'exécution Wakanda de la méthode.

Note : Si un groupe d'accès a été associé à la méthode 4D, vous devez veiller à ce que le groupe 4D Mobile soit inclus dans ce groupe.

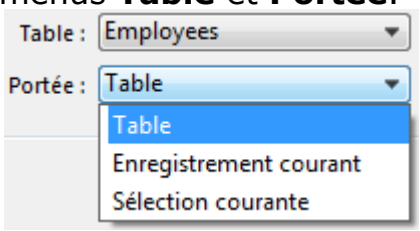
Pour définir l'exposition 4D Mobile d'une méthode projet :

1. Affichez la boîte de dialogue "Propriétés de la méthode".
   Note : Vous pouvez accéder à la boîte de dialogue des propriétés de la méthode depuis le menu contextuel de la page "Méthodes" de l'Explorateur, ou depuis le menu **Méthode/Propriétés de la méthode...** de l'Editeur de méthodes.

2. Cochez l'option **Disponible via les appels 4D Mobile** :



3. Définissez le contexte d'exécution Wakanda de la méthode projet à l'aide des menus **Table** et **Portée**.



   Ces paramétrages sont requis pour respecter la logique de Wakanda. Pour plus d'informations sur ce point, reportez-vous au paragraphe suivant.
4. Cliquez sur le bouton **OK** pour valider les modifications effectuées.
   Les méthodes projet disponibles via 4D Mobile sont listées dans la rubrique "Méthodes 4D Mobile" de l'Explorateur de 4D (cf. paragraphe "Explorateur" ci-dessous).

Table parente et Portée des méthodes projet

Lorsque vous déclarez une méthode projet disponible via les requêtes 4D Mobile, vous devez déclarer explicitement son contexte d'appel à l'aide des paramètres **Table** et **Portée**.

- **Table** : table à laquelle doit être rattachée la méthode projet. Ce paramétrage n'est pas lié directement à l'utilisation des données de la table, mais permet de désigner l'objet *datastore class* via lequel vous souhaitez accéder à la méthode via le code javascript.
  Le menu affiche la liste des tables de la base exposées en 4D Mobile. Si la méthode utilise spécifiquement les données d'une table, vous pouvez

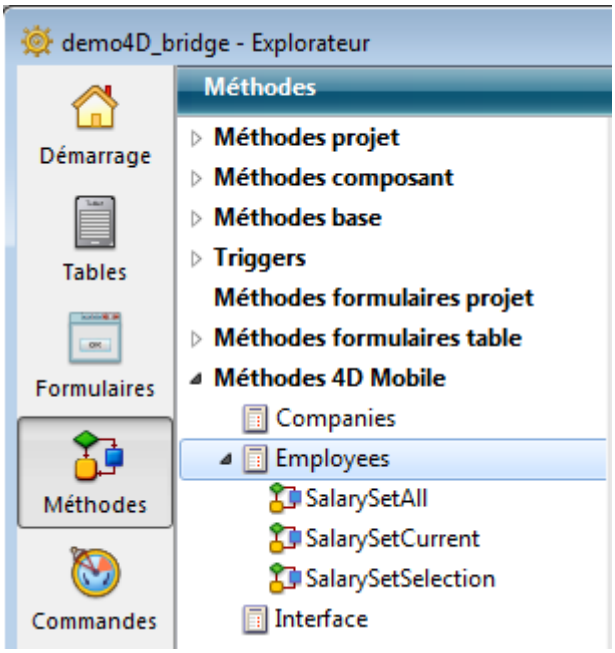
sélectionner la table. Si la méthode n'est pas liée à une seule table, vous pouvez utiliser toute table exposée. Ou encore, si vous souhaitez simplement exposer des méthodes correspondant à la logique métier de votre application 4D, vous pouvez créer et exposer une table dédiée, par exemple [Interface4DMobile], et lui associer toutes les méthodes projet exposées en 4D Mobile.

- **Portée** : périmètre d'enregistrements sur lequel sera appliquée la méthode. Cette déclaration est nécessaire car, côté Wakanda, les méthodes sont des propriétés d'objets JavaScript et ne peuvent être appelées que par l'intermédiaire de ces objets. Chaque méthode 4D exposée doit être explicitement associée au contexte de base de données dans lequel elle sera appelée : **Table**, **Sélection courante** et **Enregistrement courant**.
  - **Table** : cette option indique que la méthode 4D sera exécutée en utilisant l'ensemble des enregistrements de la table désignée.
    Côté Wakanda, la méthode devra être appelée sur un objet de type *Datastore class*, par exemple *ds.MaTable.MaMéthode*.
  - **Sélection courante** : cette option indique que la méthode 4D sera exécutée en utilisant la sélection courante des enregistrements de la table désignée.
    Côté Wakanda, la méthode devra être appelée sur un objet de type *Entity Collection*, par exemple *ds.MaTable.all().MaMéthode.*
  - **Enregistrement courant** : cette option indique que la méthode 4D sera exécutée en utilisant l'enregistrement courant de la table désignée.
    Côté Wakanda, la méthode devra être appelée sur un objet de type *Entity*, par exemple *ds.MaTable(1).MaMéthode.*

Attention : Lorsque vous modifiez l'exposition ou la portée d'une méthode projet côté 4D, il est nécessaire de recharger le modèle distant côté Wakanda afin de tenir compte des modifications effectuées.

Explorateur

Lorsque les services 4D Mobile sont activés, les tables exposées en 4D Mobile et les méthodes projet qui leur sont rattachées sont affichées dans la page "Méthodes" de l'Explorateur de 4D, rubrique **Méthodes 4D Mobile** :



Configuration de l’application Wakanda

Côté Wakanda Enterprise, la connexion à une base 4D peut s’effectuer :

- soit à l'aide la boîte de dialogue "Connect to Remote Datastore" (disponible dans Wakanda Enterprise Studio),
- soit via l’exécution d'une méthode JavaScript (**mergeOutsideCatalog()**, **openRemoteStore()** ou **addRemoteStore()**).

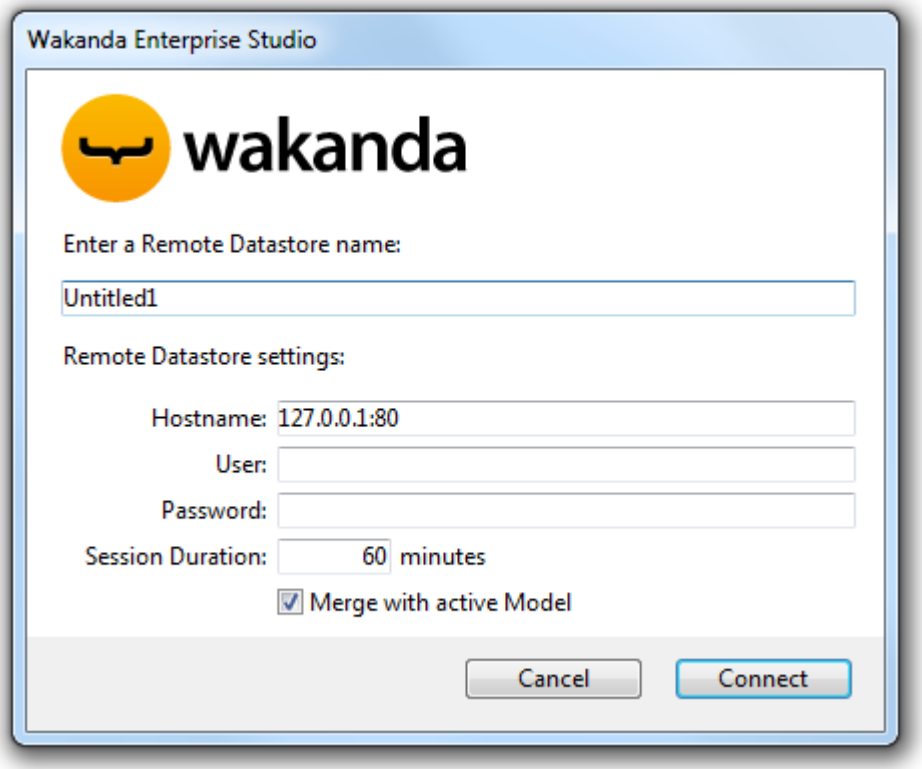
Une fois la liaison établie entre Wakanda et 4D, l’application Wakanda peut utiliser les tables, attributs et méthodes projet exposés de l’application 4D comme des objets locaux.

Il est possible d’exécuter du code JavaScript supplémentaire afin, par exemple, de modifier localement les propriétés des attributs distants, d’étendre les classes, ou encore d’ajouter des attributs calculés.

Utiliser la boîte de dialogue Connect to Remote Datastore

Dans Wakanda Enterprise Studio, la commande **Connect to Remote Datastore...** (disponible dans le menu **File** ainsi que dans le menu contextuel du projet) vous permet d'ouvrir une liaison avec un *datastore* distant. Ce *datastore* distant peut être une base 4D ou une autre application Wakanda. Dans les deux cas, le serveur HTTP du *datastore* distant doit être démarré pour que Wakanda Enterprise Studio puisse accéder au modèle distant.
Une fois la liaison définie, elle sera automatiquement rétablie à chaque ouverture de l'application en utilisant les paramètre de connexion stockés dans le fichier ".waRemoteConfig" (cf. ci-dessous).

Lorsque vous sélectionnez la commande **Connect to Remote Datastore...**, la boîte de dialogue de connexion apparaît :

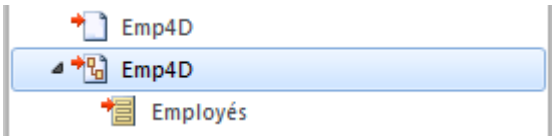


Elle contient les paramètres de connexion suivants :

- **Remote datastore name** : Nom local du catalogue distant, qui sera affiché dans l'Explorer de solution. Si vous désélectionnez l'option **Merge with active Model**, ce nom est utilisé comme identifiant de datastore au lieu de ds (cf. ci-dessous). Dans ce cas, veillez à utiliser des caractères compatibles (cf. Programming and Writing Conventions dans la documentation de Wakanda).
- **Hostname** : Adresse du serveur de données distant (utiliser HTTPS pour plus de sécurité)
- **User** et **Password** : Nom d'utilisateur et mot de passe pour l'ouverture de la session 4D Mobile sur la base 4D
- **Session duration** : Nombre de minutes (60 par défaut) pendant lesquelles maintenir la session à la base 4D distante. Ce paramètre n'est pris en compte que si la connexion est ouverte avec un utilisateur et un mot de passe non vides (la protection des accès 4D Mobile côté 4D est fortement recommandée).
- **Merge with active Model** (option cochée par défaut) : Fusionner le *datastore* distant avec le modèle actif du projet (objet **ds**) de manière à ce que les *datastore class* distantes soient incluses dans l'espace de nom **ds** et apparaissent notamment dans la liste des classes du GUI Designer de Wakanda. Pour plus d'informations, reportez-vous au paragraphe **Intégrer au modèle actif ou utiliser un modèle dédié**.

Fichiers de paramètres

Lorsqu'une connexion est établie entre Wakanda et 4D Server via la boîte de dialogue "Connect to a Remote Datastore", Wakanda Enterprise Studio crée automatiquement deux fichiers (icônes comportant une flèche rouge) dans le dossier du projet :



- le premier fichier (extension ".waRemoteConfig") stocke les paramètres de connexions définis dans la boîte de dialogue,

- le second fichier (extension ".waRemoteModel") contient la représentation locale du modèle du *datastore* distant. Son contenu peut être affiché dans la fenêtre de l'éditeur de modèles de Wakanda (mais pas modifié).

Note : Vous pouvez visualiser les extensions des fichiers dans l'infobulle qui apparaît lorsqu'ils sont sélectionnés dans l'Explorer de Wakanda Studio.

Utiliser une méthode JavaScript

Wakanda Enterprise Server vous permet d'établir une liaison avec une base 4D en exécutant une méthode JavaScript. La méthode de connexion doit généralement être placée dans le code qui s'exécute à l'ouverture de l'application (bootstrap.js) ou du modèle (model.js) afin que la liaison soit disponible lors de chaque session.

Trois méthodes vous permettent d'établir une liaison 4D Mobile :

- model.mergeOutsideCatalog()
- addRemoteStore()
- openRemoteStore()

La principale différence entre ces méthodes se situe au niveau du mode d'intégration des objets provenant du *datastore* distant dans l'application Wakanda : **model.mergeOutsideCatalog()** fusionne le catalogue distant avec le modèle actif tandis que **addRemoteStore()** et **openRemoteStore()** génèrent des modèles dédiés. Pour plus d'informations sur ce point, reportez-vous au paragraphe **Intégrer au modèle actif ou utiliser un modèle dédié** ci-dessous.

model.mergeOutsideCatalog()

La méthode JavaScript **mergeOutsideCatalog()** permet de désigner un catalogue de données distant et de le fusionner au sein de votre modèle Wakanda courant. Cette méthode doit être appelée dans le fichier .js associé au modèle courant et exécuté par le serveur Wakanda.

Deux syntaxes sont possibles :

- Syntaxe directe :

  ```
  model.mergeOutsideCatalog(nomLocal,adresse, utilisateur, motDePasse);
  ```

- Syntaxe utilisant un objet :

  ```
  model.mergeOutsideCatalog(nomLocal, {  hostname: adresse,  user: utilisateur,  password: motDePasse,  jsFile: cheminFichierJS  timeout: minutes });
  ```

L’avantage de la syntaxe avec objet est qu’elle autorise l’ajout d’un fichier .js qui sera exécuté à l’issue de la connexion à la base 4D. Ce fichier permet de modifier localement le catalogue référencé depuis la base distante.

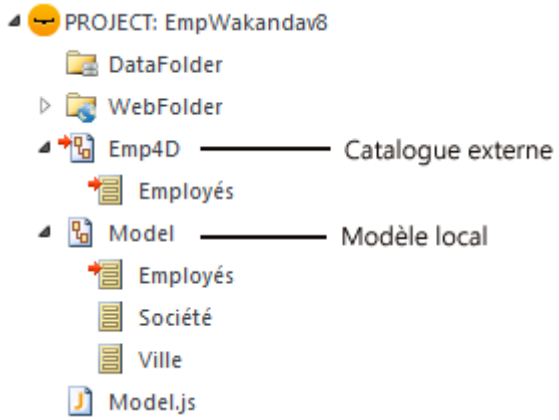
Paramètres	Type	Description
nomLocal	Chaîne	Nom local du catalogue distant
adresseIP	Chaîne	Adresse du serveur de données distant (utiliser HTTPS pour plus de sécurité)
utilisateur	Chaîne	Nom d'utilisateur pour l'ouverture de la session
motDePasse	Chaîne	Mot de passe pour l'ouverture de la session
jsFile	Chaîne	(optionnel) Chemin d'accès relatif d'un fichier JavaScript situé dans le même dossier que le modèle (cf. paragraphe **Modifier le modèle externe**)
timeout	Num	(optionnel) Timeout de connexion cliente à la base 4D en minutes (60 par défaut). A noter que ce paramètre n'est pris en compte que si la session est ouverte avec un utilisateur et un mot de passe non vides (la protection des accès 4D Mobile côté 4D Server est fortement recommandée)

Pour une description plus détaillée, reportez-vous à la documentation de la méthode mergeOutsideCatalog() dans le manuel Server-side API de Wakanda.

model

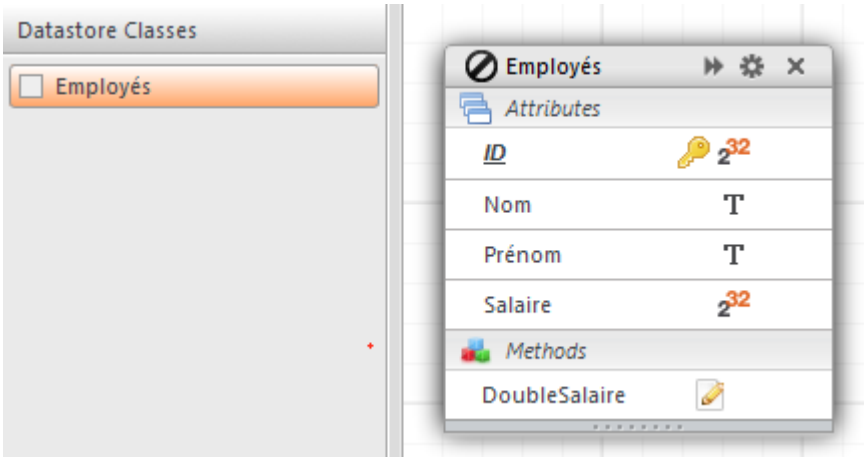
L'objet *model* désigne le "modèle" courant de l'application Wakanda, c'est-à-dire l'ensemble de ses "datastore classes" (tables) et méthodes. Dans le contexte d'une architecture 4D Mobile, le modèle Wakanda peut être vide. Si l'application Wakanda contient déjà des objets, les classes et méthodes référencées depuis l'application 4D distante sont fusionnées au modèle local lorsque vous utilisez la méthode **mergeOutsideCatalog()**.

Lorsque la connexion est établie avec succès, les tables 4D "exposées" sont ajoutées aux classes du modèle côté Wakanda. Dans Wakanda Enterprise Studio, elles apparaissent parmi les classes du modèle local, identifiées par une petite flèche rouge. Le catalogue externe est également représenté dans Wakanda Studio par un catalogue spécifique (nommé *nomLocal*.waRemoteCatalog) signalé par une petite flèche rouge :



Note: Les extensions de fichier peuvent être masquées dans Wakanda Studio.

Vous pouvez double-cliquer sur ce fichier pour visualiser le catalogue externe dans l'éditeur de modèles de Wakanda Studio :



Exemple

- Exemple de connexion directe :

  model.mergeOutsideCatalog("base4D","localhost:80", "admin", "123456");

- Exemple de connexion utilisant un objet :

  model.mergeOutsideCatalog("base4D", { hostname: "http://localhost:8050", user: "wak", password: "123456", jsFile: "base4D.js" timeout: 15 });

openRemoteStore() et addRemoteStore()

Les méthodes **openRemoteStore()** et **addRemoteStore()** constituent des moyens alternatifs d'établir des connexions dynamiques entre une application Wakanda et une application 4D.

Comme **mergeOutsideCatalog()**, ces méthodes permettent d'accéder dynamiquement aux données des bases 4D mais leur fonctionnement est différent :

- elles permettent de référencer un modèle distant à tout moment au cours de la session Wakanda -- et non au chargement de la solution.
- les tables, attributs et méthodes du modèle externes sont accessibles via un *datastore* distinct, ils ne sont pas fusionnés au modèle local de l'application Wakanda (accessible via l'objet **ds**).

openRemoteStore() retourne une référence valide uniquement dans le contexte JavaScript courant, tant que **addRemoteStore()** maintient la référence durant toute la session.

Pour plus d'informations, reportez-vous à la description des méthodes **openRemoteStore()** et **addRemoteStore()** dans la documentation de Wakanda.

Intégrer au modèle actif ou utiliser un modèle dédié

Quel que soit le mode de connexion avec le *datastore* 4D distant (boîte de dialogue "Connect to Remote Datastore" de Wakanda Studio ou exécution d'une méthode JavaScript), vous devez choisir si les classes (tables) distantes doivent être fusionnées avec le modèle actif ou si elles doivent être placées dans un modèle dédié.

Cette alternative est résumée dans le tableau suivant :

Pour...	fusionner au modèle actif	utiliser un modèle dédié
Dialogue "Connect to Remote Datastore"	Cocher *Merge with active Model*	Déselectionner *Merge with active Model*
Méthode JavaScript	mergeOutsideCatalog()	openRemoteStore() ou addRemoteStore()

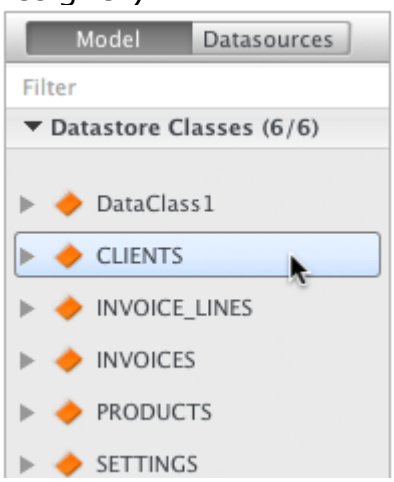
Fusionner le modèle actif

Lorsque vous fusionnez les tables 4D distantes avec le modèle actif, elles sont intégrées au modèle par défaut de l'application (dont le datastore est l'objet **ds**), comme les classes locales. Les principes d'accès aux données sont les suivants :

- côté serveur, vous accédez aux tables et méthodes 4D distantes via l'objet **ds** (cf. section **Appel des tables et des méthodes 4D**). Exemple :

  ```
  var invoiceList = ds.INVOICES.all(); //accès à la table INVOICES du catalogue par défaut
  ```

- côté client, vous bénéficiez des automatismes de la librairie WAF (*Wakanda Ajax Framework*) : les tables 4D distantes sont disponibles via les objets de haut niveau *datasources*, ou via l'API *dataprovider*, proposant des accès plus bas niveau.
- dans Wakanda Enterprise Studio, les tables de la base 4D sont listées avec les classes locales dans le Concepteur d'interfaces graphiques de Wakanda (GUI Designer) :



Ces principes facilitent le développement des applications 4D Mobile mais peuvent entraîner des conflits de noms entre les tables, notamment lorsque l'application Web fait appel à plusieurs *datastores* distants. Dans ce cas, il peut être utile de placer les éléments distants dans un modèle dédié.

Utiliser un modèle dédié

Lorsque les tables 4D distantes ne sont pas fusionnées avec le modèle actif, elles utilisent un modèle "dédié". Les classes distantes utilisent alors un espace de nommage qui est propre au *datastore* auquel l'application est connectée, elles ne sont pas accessibles dans l'objet **ds**. Il est alors possible d'utiliser simultanément plusieurs tables du même nom dans plusieurs datastores différents :

  - côté serveur, vous accédez aux tables et méthodes 4D distantes via un catalogue personnalisé dont le nom est celui que vous passez dans le paramètre de connexion **Remote datastore name** (dialogue) ou *nomLocal* (méthodes JavaScript). Par exemple, si vous avez créé une

liaison appelée "my4Dstore", vous pouvez écrire dans le code de l'application :

```
var invoiceList = my4Dstore.INVOICES.all(); //accès à la table INVOICES du datastore my4Dstore
```

Ce principe comporte toutefois des limitations dans la version actuelle de Wakanda Enterprise :

- il n'est pas possible aux applications clientes d'accéder directement aux classes distantes via la librairie WAF ou via REST,
- les classes distantes ne sont pas listées dans le *GUI Designer* de Wakanda Enterprise Studio.

Il est donc généralement conseillé de choisir le mode fusionné pour les *datastores* distants, si votre application cliente doit accéder directement aux données des tables 4D distantes.

Modifier le modèle externe

Wakanda Enterprise vous permet de modifier certaines caractéristiques de la version locale du modèle externe, dans un but de personnalisation, d'optimisation ou de sécurisation.

Pour cela, il vous suffit d'ajouter du code JavaScript approprié dans un fichier .js ayant le même nom local du catalogue avec le suffixe .js et placé dans le même dossier que le modèle. Par exemple, si le nom du catalogue local est *Emp4D.waRemoteMode*l, vous devez utiliser un fichier nommé *Emp4D.js* placé dans le dossier du modèle.

Notes :

- A compter de la version 11, par défaut ce fichier est créé automatiquement par Wakanda Studio.
- Il est possible d'utiliser un autre nom à l'aide du paramètre *jsFile* si vous utilisez une méthode JavaScript pour la connexion.

Ce fichier est exécuté par Wakanda à l'initialisation du catalogue externe. A l'aide de ce fichier, vous pouvez notamment :

- modifier les propriétés des attributs des datastore class, tels que les événements ou la portée. Exemple :

  ```
  model.nomClass.nomAttribut.scope ="publicOnServer"
  ```

- ajouter des attributs calculés aux datastore class. Exemple :

  ```
  model.nomClass.calcAtt = new Attribute("calculated", "string"); model.nomClass.calcAtt.onGet = function(); model.nomClass.calcAtt.onSet = function();
  ```

- ajouter des attributs alias aux datastore class. Exemple :

  ```
  model.nomClass.newAlias = new Attribute("alias", "number", "Link_15.cinteger");
  ```

- créer des datastore class locales dérivées des tables du catalogue externe, afin de contrôler entièrement les données envoyées aux clients. Une datastore class dérivée permet de présenter une vue personnalisée d'une table externe, tout en conservant l'accès global à la datastore class étendue (parente) sur le serveur Wakanda. Exemple :

```
model.ClasseDerivee = new DataClass("Emps", "public", "MaTable4D")
```

- supprimer des attributs des datastore class locales dérivées, par sécurité ou pour optimiser le trafic réseau. Exemple :

  ```
  model.ClasseDerivee = new DataClass("Emps", "public", "MaTable4D")
  model.ClasseDerivee.removeAttribute("salaire");
  model.ClasseDerivee.removeAttribute("commentaires"); model.ClasseDerivee.removeAttribute("...");
  ```

  Avec cet exemple, vous avez créé une classe dérivée nommée "ClasseDerivee", basée sur la classe "MaTable4D", qui n'enverra via le réseau que les attributs que vous voulez.

Pour plus d'informations sur le code JavaScript de manipulation des modèles, reportez-vous au chapitre Model API dans la documentation de Wakanda.

Définir des permissions

Vous pouvez définir des permissions spécifiques à Wakanda Server globalement pour le modèle distant et/ou individuellement pour chaque classe. Pour plus d'informations sur ce point, reportez-vous à la section Assigning Group Permissions dans la documentation de Wakanda.

Appel des tables et des méthodes 4D

Appel des tables 4D

Le mode d'accès aux tables 4D référencées dans l'application Wakanda dépend du type d'intégration du catalogue externe, défini sur Wakanda lors de la connexion à l'application distante (cf **Intégrer au modèle actif ou utiliser un modèle dédié**) :

- fusion au modèle actif (option par défaut) : dans ce cas, les tables distantes sont utilisées exactement comme les classes locales via l'objet **ds**.
- utilisation d'un modèle dédié : dans ce cas, les tables distantes sont des propriétés de l'objet model dédié.

Tables fusionnées au modèle actif

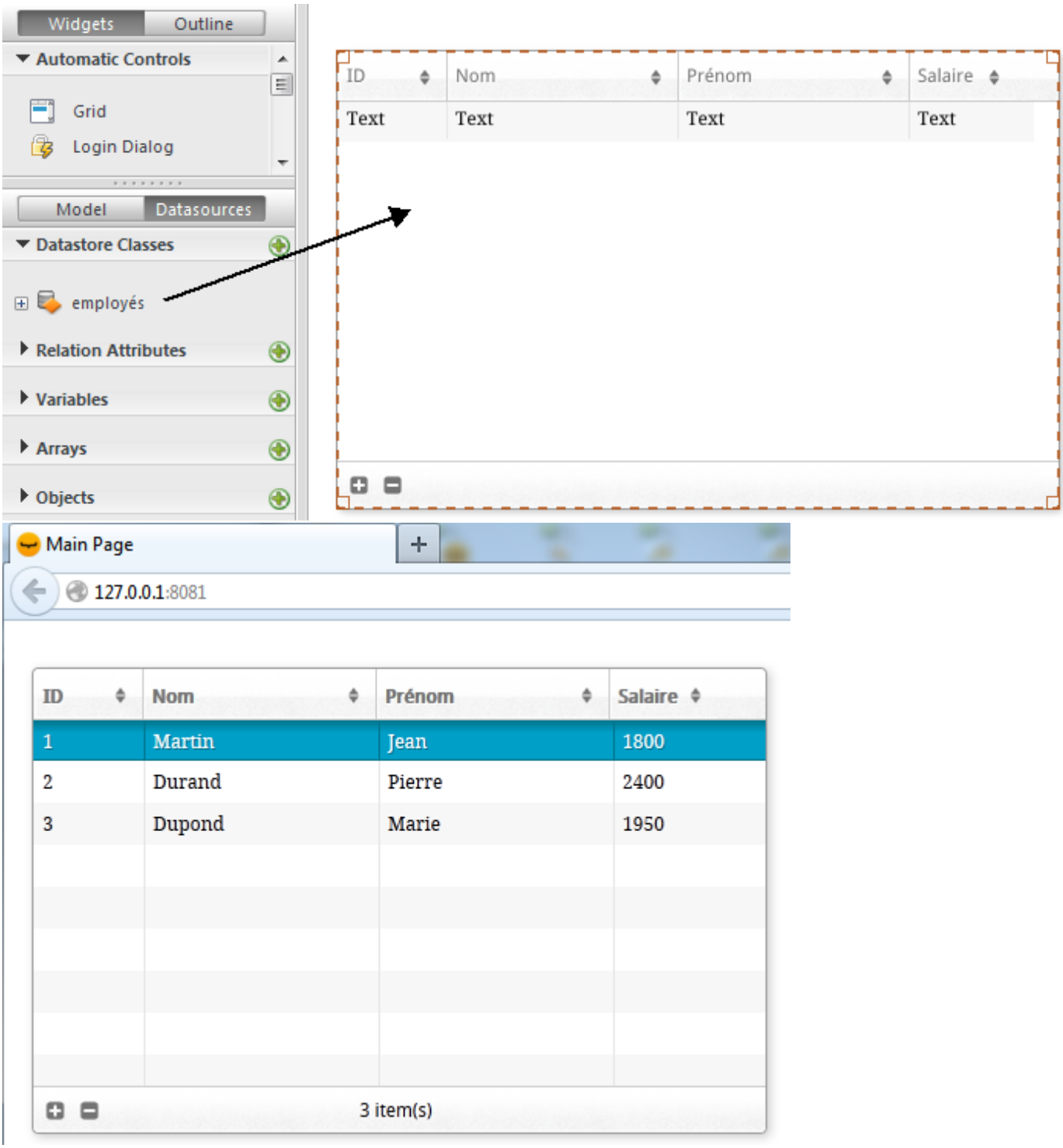
En cas de fusion au modèle actif, les tables 4D référencées dans l'application Wakanda peuvent être utilisées directement dans le code JavaScript *server-side* (SSJS) comme propriétés de l'objet **ds**, tout comme les *datastore class* locales.

Note : L'objet ds contient le *datastore* courant de l'application Wakanda.

Par exemple pour effectuer une recherche parmi les enregistrements de la table [Employees], vous pouvez écrire :

```
var emp = ds.Employees.query("age > :1",30);          //récupérer la collection des enregistrements de la table Employees          //dont l'âge est supérieur à 30 dans la variable emp
```

Côté client, vous pouvez également bénéficier des mécanismes automatiques des *datasources* basées sur des *datastore class* et associées à des widgets. Par exemple, si vous associez la datasource 'employés' à un widget de type 'Grid', vous pouvez afficher automatiquement la liste des employés :



Lorsque la table est associée à une *datasource*, vous pouvez également accéder à ses données via cette *datasource*. Par exemple pour trier la collection d'enregistrements de la *datasource* 'employés', vous pouvez écrire :

```
sources.employés.orderBy("age");      //trier la collection d'employés en fonction de l'age
```

Pour plus d'informations sur la manipulation des *datastore class*, reportez-vous à la documentation de Wakanda.

Tables placées dans un modèle dédié

Les tables 4D référencées sont utilisées dans le code JavaScript *server-side* (SSJS) comme propriétés du catalogue dans lequel elles ont été placées au moment de la création de la liaison. Le nom du catalogue est celui que vous passez dans le paramètre de connexion **Remote datastore name** (dialogue de connexion de Wakanda Studio) ou *nomLocal* (méthodes JavaScript).

Par exemple, si vous avez créé une liaison appelée "my4Dstore" et souhaitez effectuer une recherche parmi les enregistrements de la table [Employees], vous pouvez écrire :

```
var emp2 = my4Dstore.Employees.query("age > :1", 30);      // chercher parmi les enregistrements de la table Employees      // dans la liaison nommée "my4Dstore"
```

Note d'implémentation : *Côté client, l'utilisation d'un modèle dédié dans la version actuelle de 4D Mobile n'autorise pas pour l'instant l'accès aux classes distantes.*

Appel des méthodes 4D

Portée et objets

Les méthodes 4D référencées dans l'application Wakanda peuvent être utilisées directement dans le code JavaScript comme propriétés des objets **datastore class**, **entity collection** ou **entity**, en fonction de leur portée définie côté 4D (cf. paragraphe **Table parente et Portée des méthodes projet**). Voici la correspondance entre les objets Wakanda et la portée des méthodes projet :

Portée 4D	Objet Wakanda
table	datastore class
sélection courante	entity collection
enregistrement courant	entity

Note : Les méthodes 4D peuvent également être appelées côté client via des *datasources* (cf. ci-dessous), dans ce cas toutes les méthodes sont disponibles, la *datasource* les appliquant automatiquement à la collection courante ou l'entity courante en fonction du contexte.

Par exemple, si vous effectuez une recherche avec la méthode query (cf. paragraphe précédent), Wakanda retourne une entity collection. Vous pouvez exécuter sur cette collection toute méthode projet 4D dont la portée déclarée est "sélection courante".

Serveur et Client

Les méthodes 4D peuvent être appelées par du code JavaScript de trois manières :

- depuis du code JavaScript exécuté sur le serveur (SSJS), via l'API SSJS Datastore. Dans ce cas, les méthodes 4D sont appelées en tant que propriétés des objets **datastore class**, **entity collection** ou **entity**, comme décrit ci-dessus.
  Exemples :

  ```
  var vTot = ds.Emp.raiseSalary(param))      //raiseSalary est une propriété de datastore class      //le catalogue est fusionné au modèle actif var vTot2 = my4DStore.Company.first().capital(param))      //capital est une propriété d'entity car first() retourne une entity      //utilsation du modèle dédié my4DStore
  ```

- depuis du code JavaScript exécuté sur le client (c'est-à-dire le navigateur) via le Framework Ajax Wakanda (WAF). Deux possibilités s'offrent à vous :
  Note d'implémentation : *Dans la version actuelle de Wakanda Enterprise, l'accès client aux méthodes de la base 4D est disponible uniquement lorsque la base distante est connectée avec fusion au modèle actif.*
  - utiliser l'API WAF Datasource : cette API de haut niveau propose de nombreux automatismes pour gérer les données. Avec cette API, les méthodes 4D sont appelées en tant que **propriétés des datasources**

associées aux datastore class et seront automatiquement appliquées à la datastore class, l'entity collection courante ou l'entity courante en fonction du contexte. Vous pouvez gérer les valeurs de retour des méthodes ou les éventuelles erreurs en utilisant la syntaxe asynchrone (requise pour le code exécuté sur le client).
Exemple :

```
sources.employee.raiseSalary(param, {onSuccess: function(event) { ... //code à exécuter lorsque la méthode a terminé} }))
```

Il n'est pas obligatoire d'utiliser une fonction de *callback* car les objets *datasources* proposent des automatismes prenant en charge par exemple la mise à jour des données affichés dans la collection courante à l'issue d'une recherche.

- utiliser l'API WAF Dataprovider : cette API cliente de bas niveau permet de manipuler directement les objets. Comme pour l'API SSJS Datastore, les méthodes 4D sont appelées en tant que propriétés des objets **datastore class**, **entity collection** ou **entity**. Vous devez toutefois gérer les valeurs de retour des méthodes ou les éventuelles en utilisant la syntaxe asynchrone (requise pour le code exécuté sur le client). Exemple :

```
ds.Employee.raiseSalary(param, // la syntaxe ressemble à un appel SSJS {onSuccess: function(event) // mais c'est du code client, il faut gérer // la méthode callback de l'appel asynchrone { ... //code à exécuter lorsque la méthode 4D a terminé} }))
```

Le choix d'un emplacement (serveur ou client) et de l'API dépend des besoins de l'application et est détaillé dans la documentation de Wakanda.

Paramètres

Comme pour les méthodes standard, vous pouvez passer des paramètres lors de l'appel, ils seront reçus dans l'ordre dans les paramètres *$1*, *$2*, etc. De même, la méthode peut retourner un résultat dans la variable *$0*.

Exemple : Vous souhaitez augmenter de 5% les employés dont le salaire est inférieur à 1500.

- Côté 4D, la méthode projet ***AugmentSalaire*** a été exposée via 4D Mobile et sa portée est "Sélection courante". Son code est le suivant :

```
C_REEL($1)
LECTURE ECRITURE([Employees])
DEBUT SELECTION([Employees])
Tant que(Non(Fin de selection([Employees])))
   [Employees]salary:=[Employees]salary*$1
   STOCKER ENREGISTREMENT([Employees])
   ENREGISTREMENT SUIVANT([Employees])
Fin tant que
LIBERER ENREGISTREMENT([Employees])
```

- Côté Wakanda, vous exécutez le code suivant sur le serveur :

```
var emp = ds.Employees.query("salary < :1",1500); // emp contient la collection des employés dont le salaire // est <1500 emp.AugmentSalaire(1.05); //exécuter la
```

```
méthode AugmentSalaire sur la collection          //Vous pourriez également écrire
//"ds.Employees.query("salary < :1",1500).AugmentSalaire(1.05)
```

Vous pouvez également retourner une sélection 4D directement comme une collection Wakanda à l'aide de la commande **MOBILE Renvoyer sélection**. Par exemple :

```
//méthode projet FindCountries
//FindCountries( chaine ) -> objet

C_TEXTE($1)
C_OBJET($0)
CHERCHER([Countries];[Countries]ShortName=$1+"@")
$0:=MOBILE Renvoyer sélection([Countries])
```

Mise à jour du contexte 4D

Lors de l'appel d'une méthode 4D via la liaison Wakanda :

- si la méthode s'applique à une sélection (*entity collection*), celle-ci devient la sélection courante et 4D se positionne sur le premier enregistrement de la sélection sans le charger ni activer les liens. Si la sélection est vide, la commande **Numero dans selection** retournera 0 au lieu de 1.

- si la méthode s'applique à un enregistrement (*entity*), celui-ci devient l'enregistrement courant. La sélection courante est réduite à cet enregistrement et la commande **Numero dans selection** retourne 1.
  Note : Pour des raisons d'optimisation et pour éviter des verrouillages inutiles, l'enregistrement est chargé en mode lecture seule. Cependant, la table est en mode lecture-écriture et il suffit d'appeler la commande **CHARGER ENREGISTREMENT** pour forcer le chargement de l'enregistrement en lecture-écriture lorsque c'est nécessaire.

- si la méthode s'applique à une table (*datastore class*), la sélection courante et l'enregistrement courant ne sont pas affectés.

A noter qu'après l'exécution d'une méthode via 4D Mobile, le contexte 4D est réinitialisé :

- les sélections sont réduites à 0,
- les enregistrements sont dépilés et déchargés,
- les sélections et ensembles locaux au process sont détruits,
- les transactions ouvertes pendant l'exécution de la méthode sont annulées,
- les configurations de liens automatiques par champs, destinations de recherches ou recherches sur le serveur sont réinitialisées,
- les impressions sont annulées,
- les fenêtres sont fermées,
- les éventuelles connexions SQL, PHP ou HTTP sont fermées.

Erreur de portée

Vous devez veiller à ce que la portée de la méthode 4D corresponde au type d'objet Wakanda appelant, sinon une erreur "*TypeError: 'undefined' is not a function*" sera

retournée par Wakanda.

Par exemple, soit la méthode 4D "getcursel" qui contient le code suivant:

```
$0:=Enregistrements trouves([Table_1])
```

Soit la méthode *run* côté Wakanda :

```
var tt = ds.Table_1.query("Field_2 = 'a*'").getcursel();
```

La méthode **query( )** retourne une collection. Si la portée de la méthode *getcursel* a été définie comme "Enregistrement courant", Wakanda retournera l'erreur suivante : *TypeError: 'undefined' is not a function (evaluating 'ds.Table_1.query("Field_2 = 'a*'").getcursel()')"*.

Exploitation des liens

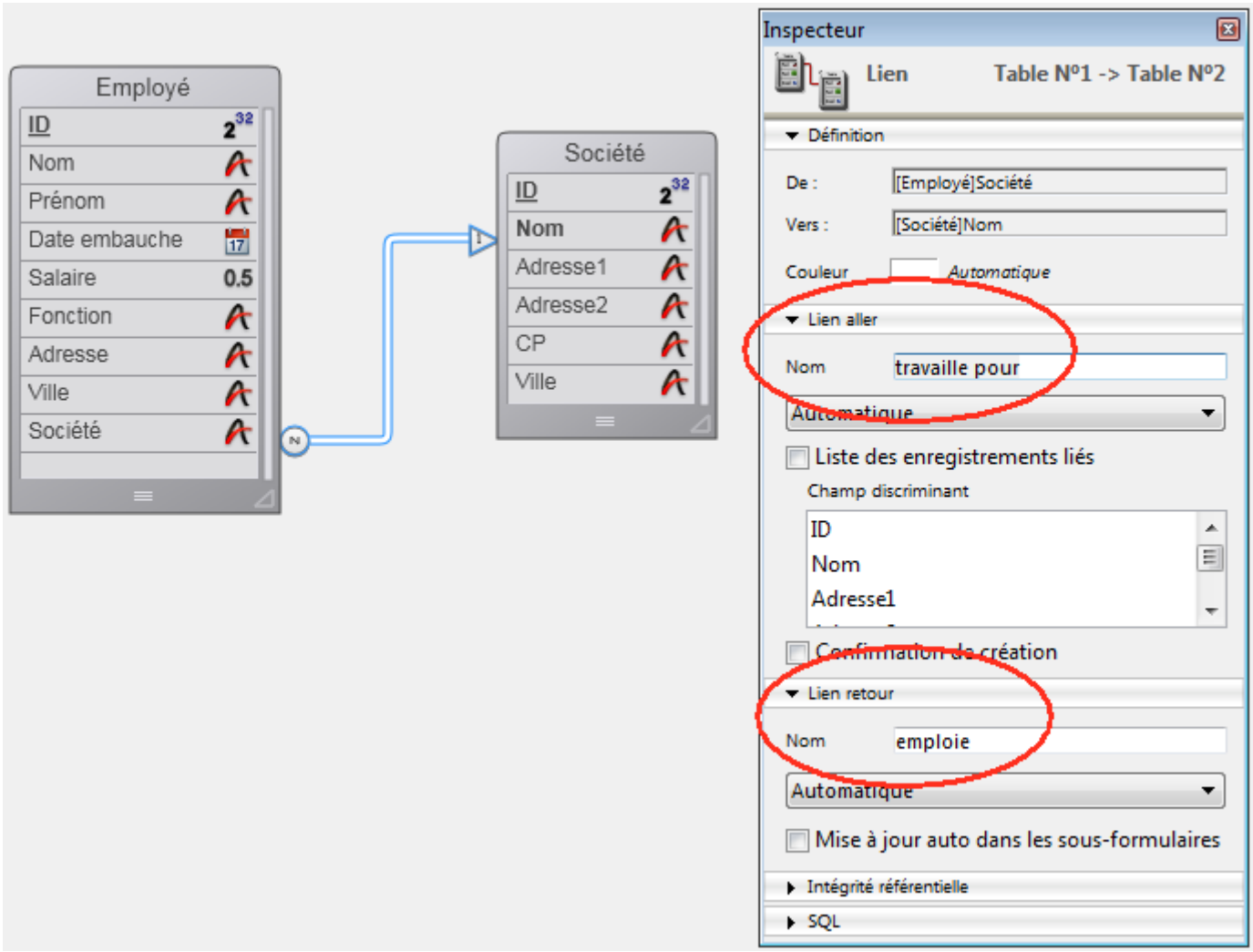
Les relations établies entre les tables 4D sont exploitées de manière transparente dans une liaison 4D Mobile. En revanche, la représentation de ces relations diffère dans Wakanda au niveau du modèle. Dans l'éditeur de modèles, les liens sont rattachés à des attributs spécifiques, appelés *attributs relationnels*. Ces attributs peuvent être utilisés directement pour afficher des données liées ou effectuer des requêtes. Pour plus d'informations sur ce point, reportez-vous à la section "Attributes" dans la documentation de Wakanda.

Chaque lien établi côté 4D entraîne l'ajout de deux attributs relationnels dans la représentation du modèle côté Wakanda :

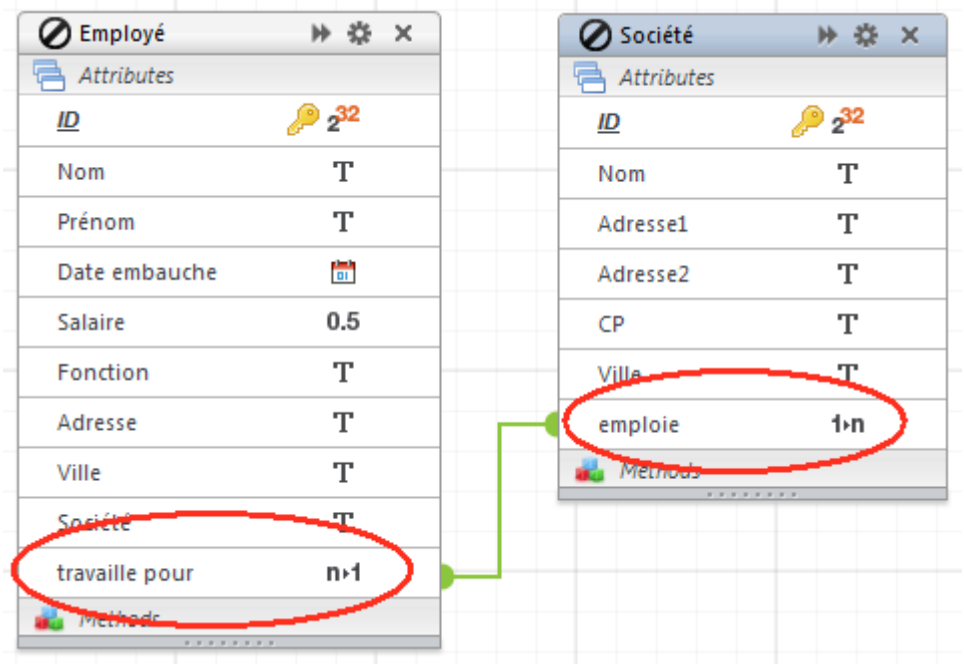
- un attribut n->1 dans la table (classe) d'où part le lien
- un attribut 1->n dans la table (classe) où arrive le lien

Ces deux attributs prennent le nom du lien tel qu'il a été défini respectivement pour le lien aller et le lien retour dans l'Inspecteur côté 4D.

Par exemple, imaginons que dans le cadre d'une structure classique "Employé/Société" vous créez un lien de la table [Employé] vers la table [Société]. Vous pouvez caractériser cette relation au travers du nom du lien : par exemple, vous pouvez nommer le lien aller "*travaille pour*" et le lien retour "*emploie*" :



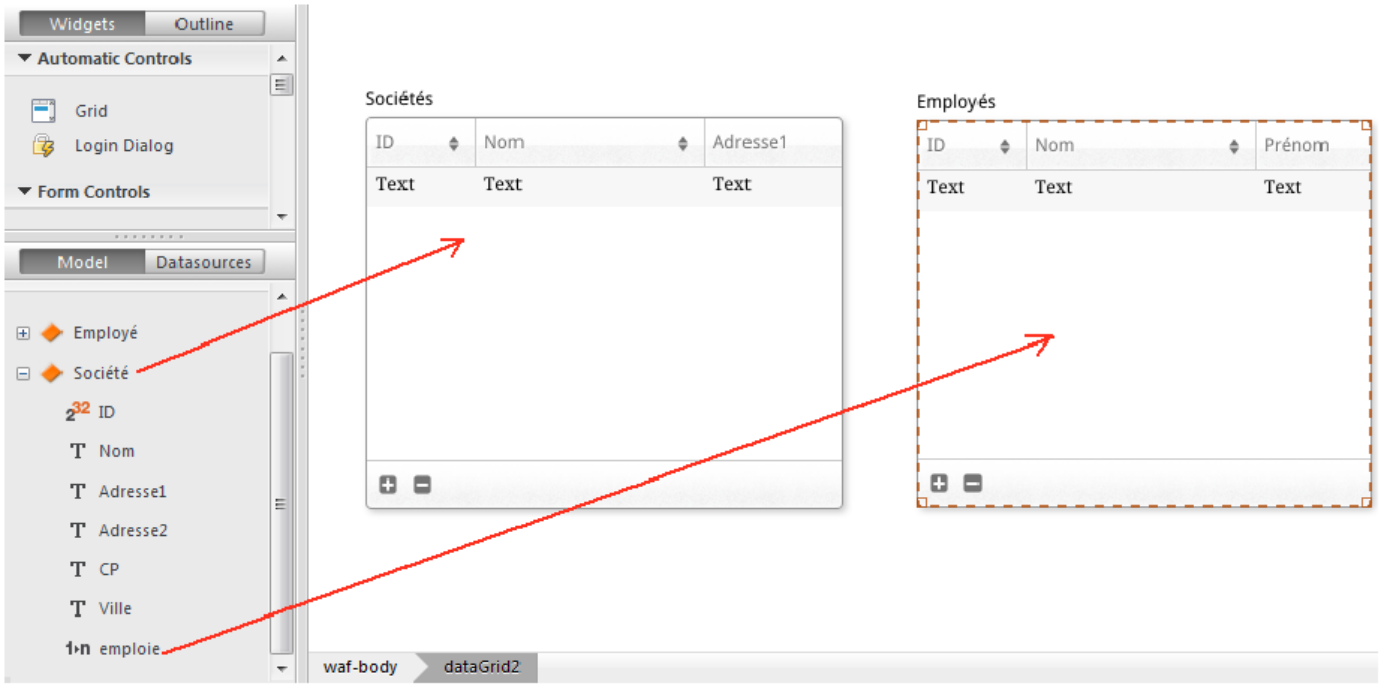
Côté Wakanda, dans le cadre d'une liaison via le connecteur pro, ces liens sont automatiquement matérialisés par deux attributs relationnels supplémentaires, que vous pouvez visualiser dans l'éditeur de modèle :



Vous pouvez nommer ces liens et donc les attributs relationnels comme vous le souhaitez, en fonction de la logique de votre application.

L'intérêt de ce principe est qu'il est très simple côté Wakanda d'utiliser ces attributs afin de manipuler les données liées. En particulier, vous pouvez créer des widgets associés à des *datasources* basées sur attributs relationnels. Ces widgets sont alors automatiquement gérés et mis à jour en fonction des actions de l'utilisateur.

Par exemple, vous pouvez très facilement créer une page contenant une grille avec la liste des sociétés et une autre contenant la liste des employés de la société sélectionnée. Pour cela, vous associez la *datastore class* "Société" à une grille et l'attribut relationnel "emploie" à l'autre grille :



Les *datasources* correspondantes sont automatiquement créées et en exécution, les deux grilles sont automatiquement synchronisées :

Sociétés

ID	Nom	Adresse1
1	Larmor SA	10 rue des fleu
2	Au bon bois	2 place de la se
3	4D SAS	62 rue d'Alsace
4	Factory Mode	20 rue du senti
5	COGIP	Place de la bata

5 item(s)

Employés

ID	Nom	Prénom
4	Robert	Marie
8	Gilles	Hermant
9	Francis	Lemonnier
10	Christian	Noe

4 item(s)

Gestion des sessions 4D Mobile

Présentation

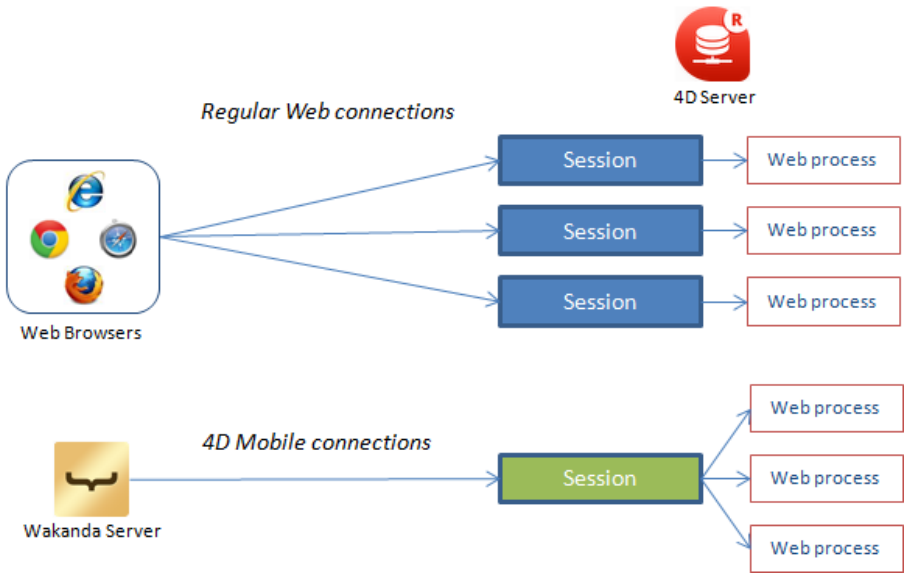
A compter de 4D v15 R4, il est possible d'accéder par programmation à l'ID d'une session 4D Mobile sur 4D Server. Cette fonctionnalité permet aux développeurs de lire ou d'écrire localement des informations liées aux sessions (cf. exemple ci-dessous).

Les sessions 4D Mobile sont gérées à l'aide des commandes standard de session Web de 4D. Plusieurs commandes Web 4D, la commande **WEB Lire nombre process session** ainsi que la **Méthode base Sur fermeture process Web** prennent directement en charge les sessions 4D Mobile.

Pourquoi et comment gérer une session 4D Mobile?

Les sessions 4D Mobile et les sessions Web sont deux sortes de sessions différentes. Bien qu'elles partagent certains concepts (et commandes),elles n'ont pas les mêmes propriétés. La principale différence a trait à la relation entre une session, un process et le contexte du process :

- Une session Web est liée à un seul process Web ; grâce à la fonctionnalité de Gestion automatique des sessions, le contexte du process (instances de variables, sélections...) de la session peut être réutilisé
- Une session 4D Mobile peut être liée à plusieurs process Web ; chaque contexte de process est automatiquement réinitialisé à l'issue de l'exécution de la méthode du process



Par conséquent, le partage d'informations liées à la session entre les process Web 4D Mobile requiert des implémentations spécifiques sur 4D Server.

Commandes 4D utilisables

Les commandes suivantes de gestion des sessions Web prennent en charge les sessions 4D Mobile.

WEB FERMER SESSION (idSession)

La commande **WEB FERMER SESSION** referme la session 4D Mobile dont l'ID a été passé dans *idSession*. Comme une session 4D Mobile peut gérer plusieurs process, cette commande demande en fait à tous les process Web liés à la session de terminer leur exécution.

WEB Lire ID session courante -> ID session

La commande **WEB Lire ID session courante** retourne l'UUID associé à la session 4D Mobile courante.

WEB LIRE EXPIRATION SESSION (idSession ; dateExp ; heureExp)

La commande **WEB LIRE EXPIRATION SESSION** retourne les informations relatives à l'expiration du cookie d'une session 4D Mobile.

Le même cookie est utilisé pour tous les process d'une session 4D Mobile.

WEB Lire nombre process session (idSession) -> Nombre de process

La commande **WEB Lire nombre process session** vous permet de connaître le nombre de process existants liés à une session donnée.

- Pour une session Web "classique", la commande retourne toujours 1 (une session Web = un process),
- Pour une session 4D Mobile, la commande retourne tous les process Web qui lui sont rattachés. Cette commande est utile dans ce contexte par exemple pour exécuter une boucle parmi tous les process d'une session 4D Mobile.

Méthode base Sur fermeture process Web

La **Méthode base Sur fermeture process Web** est appelée par 4D à chaque fois qu'un process Web est sur le point de terminer son exécution. Elle prend pleinement en charge les process des sessions 4D Mobile : dans ce contexte, elle est appelée pour chaque process Web refermé, vous permettant de sauvegarder tout type de donnée (variable, sélection, etc.) générée par le process de session 4D Mobile.

Note : Pour les sessions Web "classiques", la **Méthode base Sur fermeture process Web** est appelée à chaque fois qu'une session Web, c'est-à-dire qu'un process unique de session Web, est refermé.

Exemple

Si vous souhaitez partager ou réutiliser des informations entre plusieurs process d'une même session 4D Mobile, vous pouvez utiliser l'UUID de la session 4D Mobile pour identifier les données relatives à cette session. Par exemple, après avoir effectué une recherche parmi des enregistrements, vous souhaitez conserver une sélection temporaire sur 4D Server afin qu'une requête REST ultérieure dans la même session puisse accéder directement à cette sélection. Vous pouvez écrire :

```
//créer une sélection interprocess dont le nom contient l'UUID de session
COPIER SELECTION([Emp];"<>EmpSel"+WEB Lire ID session courante)
```

```
//plus tard, vous pouvez réutiliser cette sélection depuis la même session
UTILISER SELECTION([Emp];"<>EmpSel"+WEB Lire ID session courante)
```

A propos de la sécurité des applications 4D Mobile

Une fois les données des tables de la base 4D exposées en 4D Mobile et intégrées au catalogue de l'application Wakanda, vous devez restreindre l'accès à certaines ressources "sensibles".

A la différence des applications 4D, les applications Web ne permettent pas de contrôler via l'interface les données exposées : par exemple, il ne suffit pas de ne pas afficher un champ dans un formulaire pour le rendre réellement inaccessible à l'utilisateur. Les requêtes HTTP et l'utilisation du JavaScript permettent aux utilisateurs malveillants d'obtenir potentiellement toute information d'un serveur Web insuffisamment protégé.

Le propos de cette section n'est pas de lister toutes les mesures de sécurité à prendre dans les applications 4D Mobile mais de fournir les pistes permettant de sécuriser de manière minimale les données exposées.

- **Protection des accès via 4D Mobile à la base 4D** : vous devez contrôler les demandes de connexion 4D Mobile (via REST). Vous pouvez utiliser au choix :
  - les mots de passe 4D (cf. **Contrôles automatiques par mot de passe 4D**),
  - la **Méthode base Sur authentification 4D Mobile**.

- **Contrôle de l'exposition 4D Mobile côté 4D** : chaque table, attribut et méthode peut être ou non exposé(e) en 4D Mobile. N'exposez que les données et méthodes strictement nécessaires, il est par exemple inutile d'exposer des champs non utilisés.

- **Protection des données exposées** : vous devez utiliser les systèmes de sécurité proposés par Wakanda pour contrôler le contenu accessible via les navigateurs. Plusieurs moyens (non exclusifs) s'offrent à vous :
  - **Régler la portée** (*scope*) des attributs et des méthodes provenant de la base 4D. En particulier, vous pouvez leur affecter la portée **Public on Server**, ce qui signifie que leur accès est libre pour le code s'exécutant sur le serveur, mais qu'ils ne sont pas accessibles sur les clients Web. Ce paramétrage s'effectue dans le fichier .js de configuration du modèle externe (cf. paragraphe **Modifier le modèle externe**).
  - **Utiliser les attributs calculés** : les attributs calculés se manipulent comme des attributs standard mais leurs valeurs sont retournées par des fonctions spécifiques (**onGet**, **onSet**...) exécutées lors des accès aux champs (événements). Ce principe permet de ne pas exposer les champs de la base 4D, mais uniquement les attributs calculés nécessaires. Les accès aux champs 4D sont effectués de manière sécurisée depuis le serveur Wakanda. Vous pouvez ajouter des champs calculés dans le fichier .js de configuration du modèle externe (cf. paragraphe **Modifier le modèle externe**). Pour plus d'informations, reportez-vous à la page Attributes de la documentation de Wakanda.

- **Combiner datastore class étendues et requêtes restrictives** (restricting queries): ce concept puissant permet de contrôler non seulement les attributs exposés mais également les données qu'ils peuvent afficher. Etendre une *datastore class* signifie en créer une copie (la classe dérivée) que vous pouvez altérer en lui ajoutant des attributs calculés ou en supprimant des attributs. Vous pouvez lui associer une *restricting query* : dans ce cas, tous les accès aux données de la classe dérivée déclenchent automatiquement cette requête, qui retourne uniquement les enregistrements correspondant au(x) critère(s). Ce principe vous permet de lier les données à l'utilisateur connecté au serveur Wakanda. Par exemple, dans le cadre d'une base commerciale, la requête retourne tous les clients rattachés au commercial courant. Bien entendu, seule la classe dérivée sera accessible aux clients Web.
  Vous créez des *datastore class* étendues et ajoutez des *restricting queries* dans le fichier .js de configuration du modèle externe (cf. paragraphe **Modifier le modèle externe**). Pour plus d'informations, reportez-vous à la page Programming Restricting Queries de la documentation de Wakanda.

Note : La prise en charge des requêtes restrictives dans 4D Mobile requiert au minimum la configuration suivante :
- 4D et 4D Server **v14.1**
- Wakanda Enterprise Server **v8**