4D Mobile

4D SAS によって開発されたWakandaは、JavaScriptやHTML5といった標準のテクノロジーに基づいたWebアプリケーションを開発・配付するためのプラットフォームです。
"4D Mobile" アーキテクチャーを用いれば4D-Wakanda間にダイレクトなリンクを設定することができます。この場合、最新世代のWakandaのWebインターフェースの豊富なグラフィックと機能を、4Dデータベースの実力と組み合わせて使用することができます。

もし4DとWakandaの最初のリンクをすぐに設定したいのであれば、システム要件 にある適切な環境と設定があることを確認の上、ステップバイステップ形式での解説 を参照して下さい。

4D Mobile アーキテクチャー

システム要件

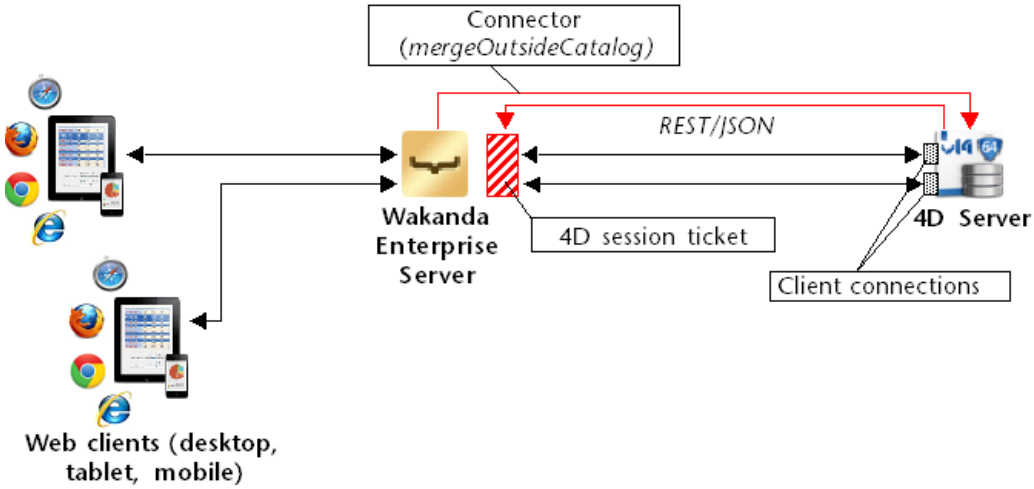
4D/Wakandaコネクターを使用してアーキテクチャーを設定する場合、以下のシステム要件が最低限必要になります:

- **4D Server v14**
  ただし4D Mobileコネクターを使用してソリューションを開発・テストしたい場合には、シングルユーザーの **4D v14** (Developer Professional)をお使いいただくことも可能です(この場合にはクライアント接続が同時に三つまでご使用いただけます)。
- **Wakanda Enterprise Server v7** と **Wakanda Enterprise Studio v7**
  開発に必要になります。どちらもWakanda download page *(Enterprise* タブ*)*からダウンロードすることができます。
- 相互に通信をするための4DデータベースとWakandaアプリケーション

4D側では、Wakandaアプリケーション側で利用したい全てのテーブル、属性、そしてメソッドを、アクセス可能な状態にしておく必要があります(4D データベースの設定 を参照して下さい)。

詳細

4D Mobile アーキテクチャーは以下の様に表すことが出来ます:



Wakandaソリューションが開始したときにWakanda サーバーによって**mergeOutsideCatalog()** JavaScript メソッドが実行されると(詳細は mergeOutsideCatalog() メソッドの実行 を参照して下さい)、Wakanda Enterprise Server と 4D Serverの間にリンクが構築されます。接続が4D Serverによって認証されると (REST 接続の管理 を参照して下さい)、RESTセッション「チケット」がWakanda サーバーへと渡されます。このチケットはWakandaによって以降の全てのRESTクライアントリクエストに対して使用されます。

このリンクによって、Wakandaサーバーは4Dデータベースの二種類のリソースにアクセスすることが出来るようになります:

- テーブルとその属性(データも含む)
- プロジェクトメソッド

認証が完了すると、これらのリソースはWakandaアプリケーションのローカルのカタログに入っていたかのように、Wakadan側にて直接使用されます(この際の接続はWakandaアプリケーションに対しては透過的です)。

WebクライアントがWakandaサーバーへ4Dデータベースへのアクセスを必要とするリクエストを送った場合、このリクエストはカレントのチケットを使用して4Dサーバーへと送られ、4D Serverマシンでは標準のクライアント接続が開かれます。この接続はデフォルトで60分のタイムアウトを上限として、ユーザがリクエストを送り続ける限り開いたままになります。このデフォルトのタイムアウト時間は **mergeOutsideCatalog()** メソッドの *timeout* 引数を使用して実行することによって変更することができます。セッション中に、4Dサーバーにて認証されたクライアント接続数がライセンス数に達してしまった場合、エラーメッセージがWakandaサーバーに返されます。

注: addRemoteStore() と **openRemoteStore()** メソッドを使用して、Wakandaと4Dアプリケーションとの間に動的で一時的なリンクを構築することもできます。これらのメソッドに関しては openRemoteStore() と addRemoteStore() を参照して下さい。

ステップバイステップ形式での解説

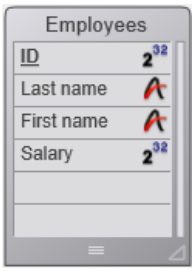
このセクションでは、ステップバイステップ形式で一つずつ手順を追って Wakanda / 4D コネクターの機能を紹介していきます。具体的には以下ような手順を解説します:

- 4D データベースの作成と設定
- 単一のページのWakandaアプリケーションの作成
- 4D データベースからのデータをWakandaのページに表示する

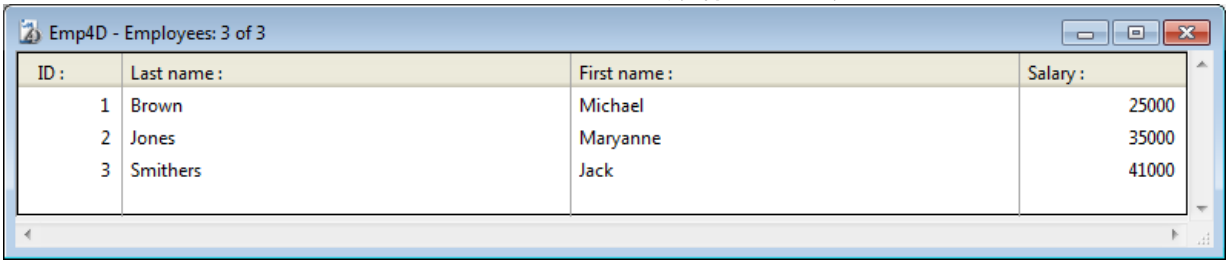
解説を簡単にするために、ここでは4D アプリケーションとWakanda アプリケーションが同じマシン上にある場合を考えていきます。もちろん、リモート構造を使用することも可能です。

1-4D データベースの作成と設定

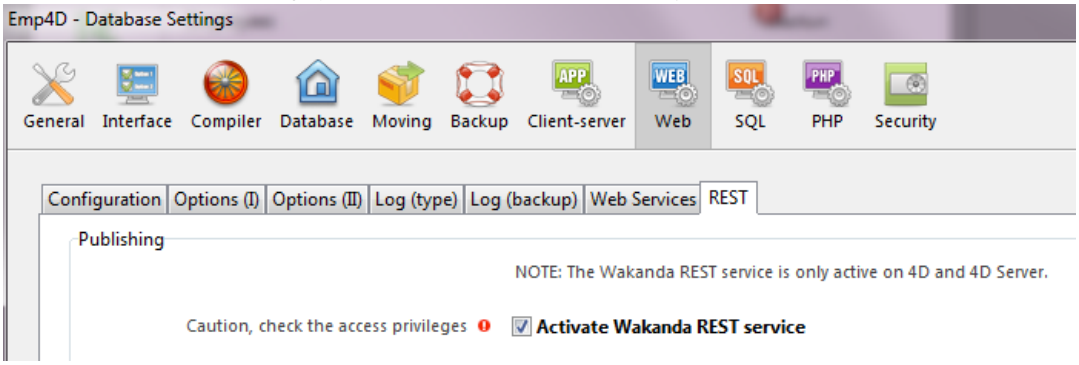
1. 4D アプリケーションまたは4D Server アプリケーションを起動して新規にデータベースを作成します。
   ここでは"Emp4D"という名前をつけたという仮定で解説を進めます。
2. ストラクチャーエディターの中で、[Employees]というテーブルを作成して以下のフィールドを追加します:
   - LastName (文字列)
   - FirstName (文字列)
   - Salary (倍長整数)

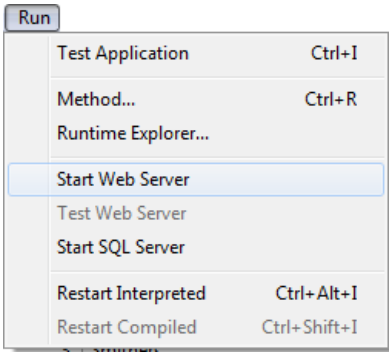
   "RESTサービスで公開"の属性は、全てのテーブルにおいて最初からチェックがされてあります。この設定は変更しないで下さい。
3. **Tables** のボタンをクリックして4Dにデフォルトフォームを作成させたのち、実際のデータを数レコード分作成します:

4. データベース設定ダイアログボックスの中から"**Web**"のページの中を表示させ、"**REST**"タブをクリックします。
5. "Wakanda REST サービスを有効化"のオプションをクリックしてオンにし、**OK**をクリックします:

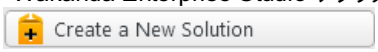
   

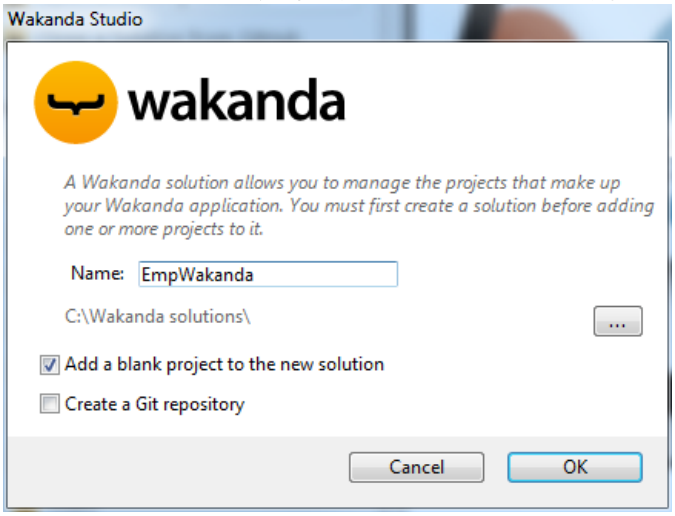
6. **実行**のメニューの中から**Web サーバー開始**を選択します:



これで4DデータベースはWakanda からのRESTリクエストに応答する準備が出来ました。なお、ここでは簡略化のためにREST接続の管理まではしていないという点に注意して下さい。実際の製品やオープンアーキテクチャの場合はREST接続を安全に管理することが必要不可欠となります(詳細な情報に関しては4D Mobileのセキュリティについて を参照して下さい)。

2 -Wakandaアプリケーションの作成

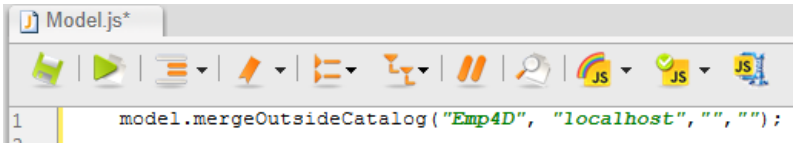
1. "Wakanda Enterprise Studio"アプリケーションを起動し、**Create a New Solution** ボタンをクリックします:


2. 作成ダイアログボックスにて、名前を記入して **OK**をクリックします。ここでは"EmpWakanda"という名前をつけて説明を続けます:


アプリケーションプロジェクトが作成され、Wakanda Studio Explorerのデフォルトの項目がウィンドウの左側に表示されます。
3. **Model** をダブルクリックします。

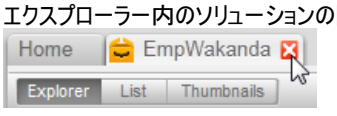

デフォルトのデータストアクラスを内包した Wakanda Model Designer が表示されます。しかしここでは外部モデルを作成したいので、このエディターは使用しません。
4. Wakanda のエクスプローラーエリアにて **Model.js** ファイルをダブルクリックします。
このファイルはGUIエディターを開いたときに自動的に作成されます。ここにはモデルの定義がJavaScript コードで記述されています。
5. 以下のJavaScript文を入力します:

```
model.mergeOutsideCatalog("Emp4D", "localhost","","");
```



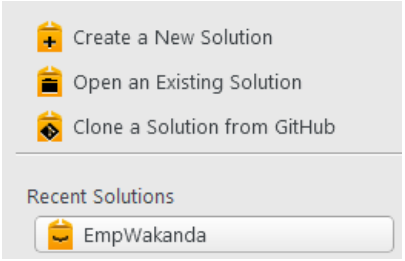
このコードはコネクターを有効化し、Wakanda プロジェクトと4Dの間のリンクを開きます:
   - "Emp4D" はWakanda Enterprise Studio 側で表示されるローカル名です。ここには任意の名前を入力できますが、ここでは簡略化のために4Dデータベースの名前を使用します。
   - "localhost" は 4D v14 のHTTPサーバーのアドレスです(必要であればホスト名またはIPアドレスを入力して下さい)。
   - 残り二つの空の文字列には名前とパスワードを入力しますが、この例では使用しないので空欄のまま話を進めます。

   ソリューションを一度閉じた後再度開き、Wakanda Enterprise Studio にモデルをロードします。

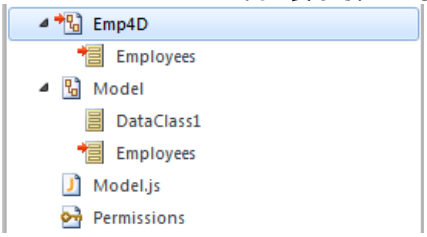
6. エクスプローラー内のソリューションの閉じるボタンをクリックします:


モデルファイルをまだ保存していない場合、ダイアログボックスが表示されます。変更を全て保存するためには **Save All** をクリックして下さい。

7. "Recent Solutions" のリスト内からソリューションをクリックして再度 Wakanda Enterprise Studio で開きます。



すると、"Emp4D" 外部モデルがWakandaアプリケーションのファイル内に表示され、4Dアプリケーションの [Employees] テーブルがローカルモデルの datastore classes 内に表示されているのが確認できます。外部モデルは赤い矢印付で表示されています:



うまく行かない場合は...

この段階でリストにテーブルが表示されていないのであれば、以下の点をチェックして下さい:

- サードパーティサービスやソフトウェア(例えばインスタントメッセンジャーなど)が4D HTTP サーバーの公開ポートと競合していないか(初期設定値では80)。
- 4D側で、4D Webサーバーが開始され、Wakanda RESTサービスが有効化されていて、テーブルがRESTへ公開されているか。
- **mergeOutsideCatalog()** へ渡されたアドレスが有効であるか。

4D Serverが実際にRESTリクエストに反応しているかどうかを調べるためには、以下のURLをブラウザに入力して下さい:

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

(RESTに公開されている全てのテーブルを返します。)

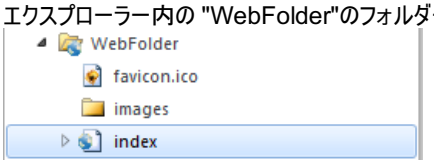
```
<address>/rest/my_table/my_method
```

(メソッドが結果を返すならば、その結果を全て返します。)

3 - Wakanda ウィジェットを使用して4D dataを表示する

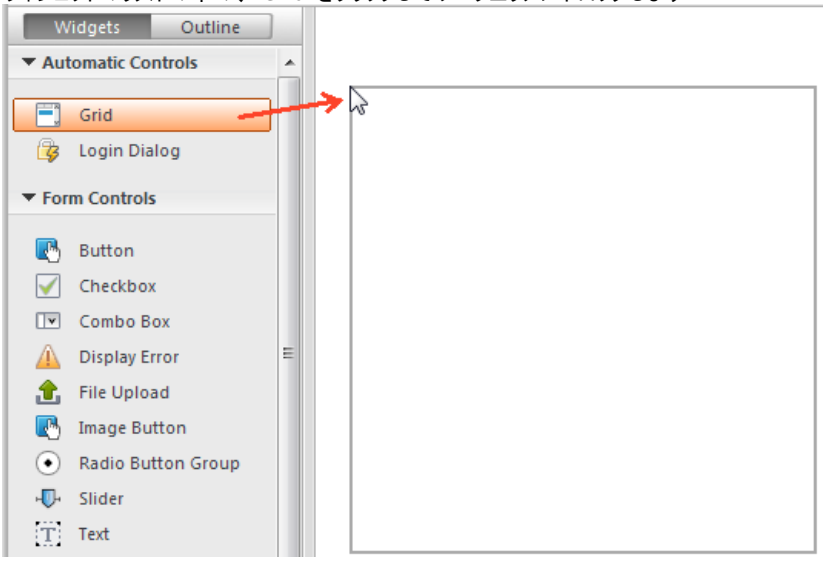
ここでは単純なドラッグ&ドロップによって4DテーブルとWakandaウィジェットを関連づけ、Wakanda Enterprise Server を起動してデータを表示させます。

1. エクスプローラー内の "WebFolder"のフォルダーを開き、Index のページをダブルクリックし、WakandaのGUIデザイナーを開きます。

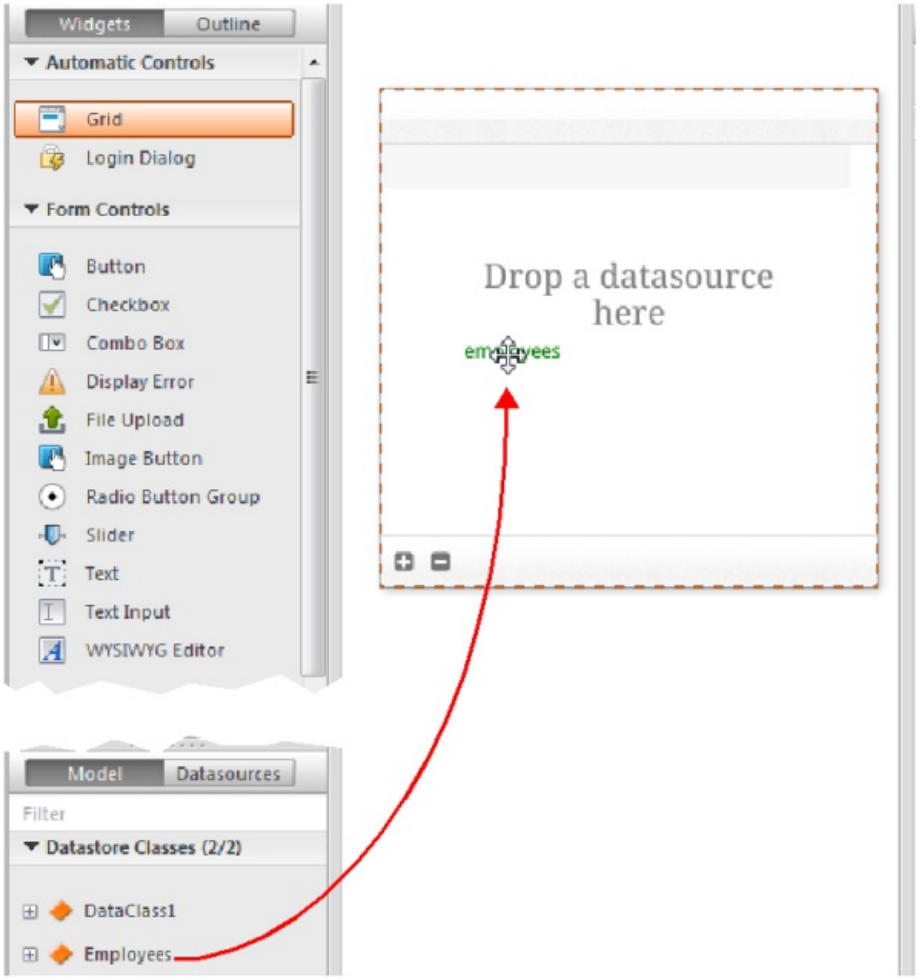


注: "WebFolder"にはプロジェクトの中のWeb 公開に必要な要素が置かれています。"Index"はプロジェクトのデフォルトのページになります。

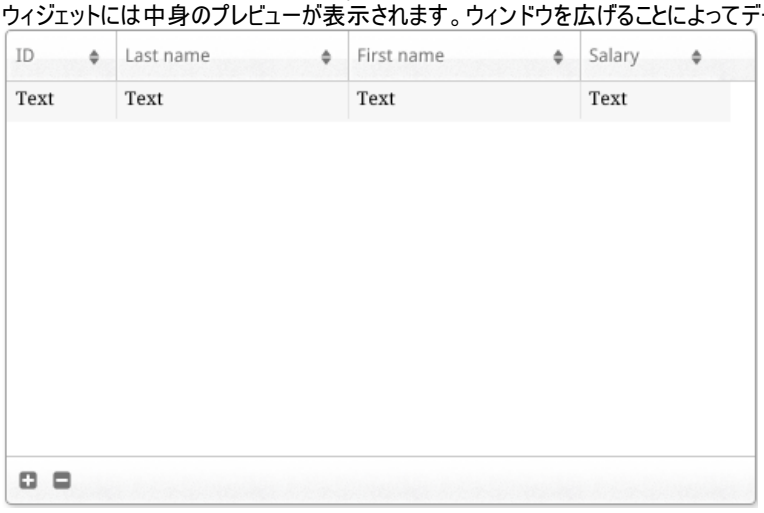
2. ウィジェットのリストの中の、"Grid"をクリックしてワークエリアにドロップします:



3. モデルのDatastore Classes のリスト内の"Employees"をクリックし、作成したグリッドの中にドロップします:

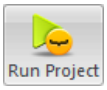


この時点で、エディターは"Employees"クラスをもとにした datasource を自動的に作成します。これはウィジェットのコンテンツを管理します。この *datasource* とはWakanda によって管理される JavaScript オブジェクトで、デフォルトでは"employees"という名前がついています(クラス名の頭文字が小文字になったものです)。
ウィジェットには中身のプレビューが表示されます。ウィンドウを広げることによってデータソースの全フィールドを表示することができます。

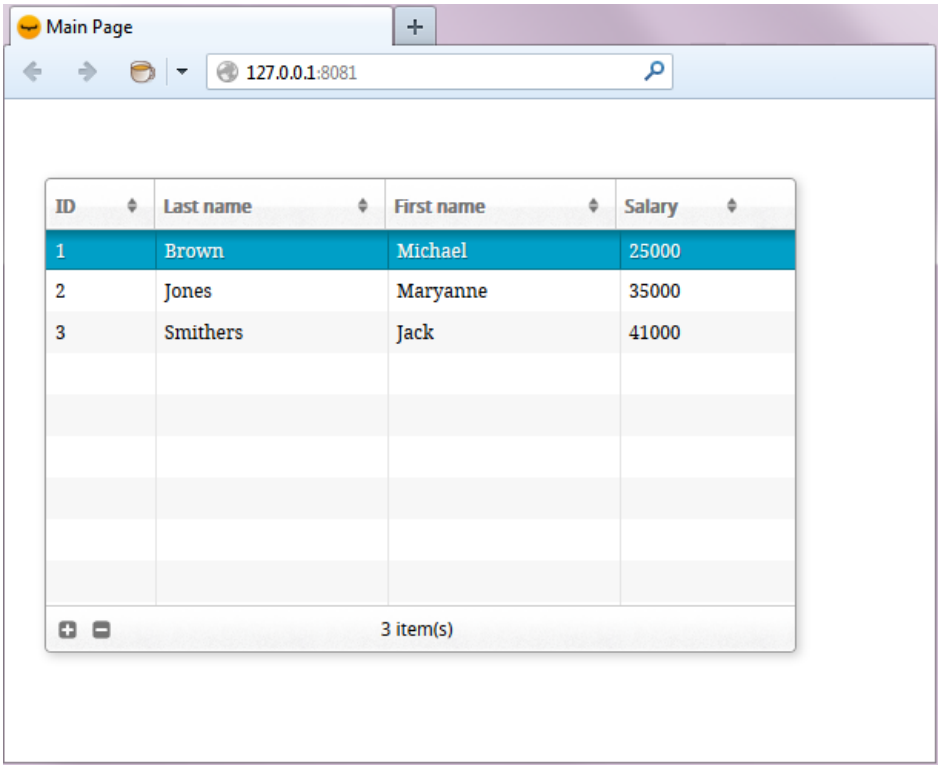


これによって*datasource* とウィジェットの関連付けが完了しました。

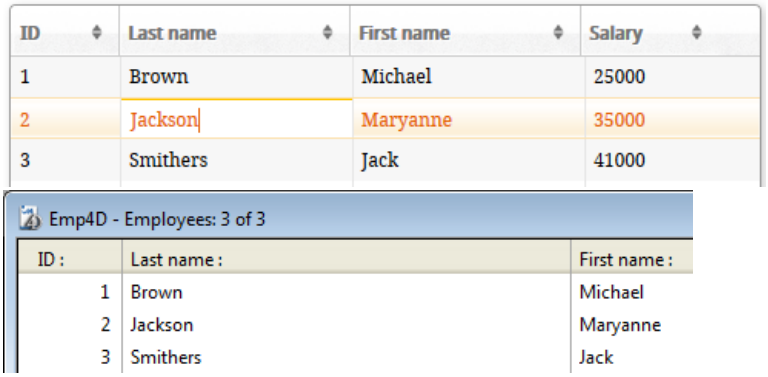
4. エディターのツールバーの **Save**  ボタンをクリックします。
   今度はブラウザを使用してデータを表示させてみましょう。
5. Wakanda Enterprise Studioのツールバーの **Run project** をクリックします:



これをクリックすることにより Wakanda Enterprise Server が開始し、"EmpWakanda"アプリケーションをパブリッシュします。先に設定しておいた4D Mobile リンクのおかげで、4D データベースのデータを既定のブラウザのウィンドウ内に表示させることが出来ます:



Web側でデータを変更することによってリンクのダイナミックな性質をテストすることもできます。例えば、ここでは Maryanne Jones' の名字を"Jackson"に変えたのが、4D側でも直ちに反映されています:



4 - 4D メソッドの作成と呼び出し

ここではとても単純なプロジェクトメソッドを4D側で作成し、Webページ側から実行します。このメソッドは全てのsalaryの値を二倍にします。

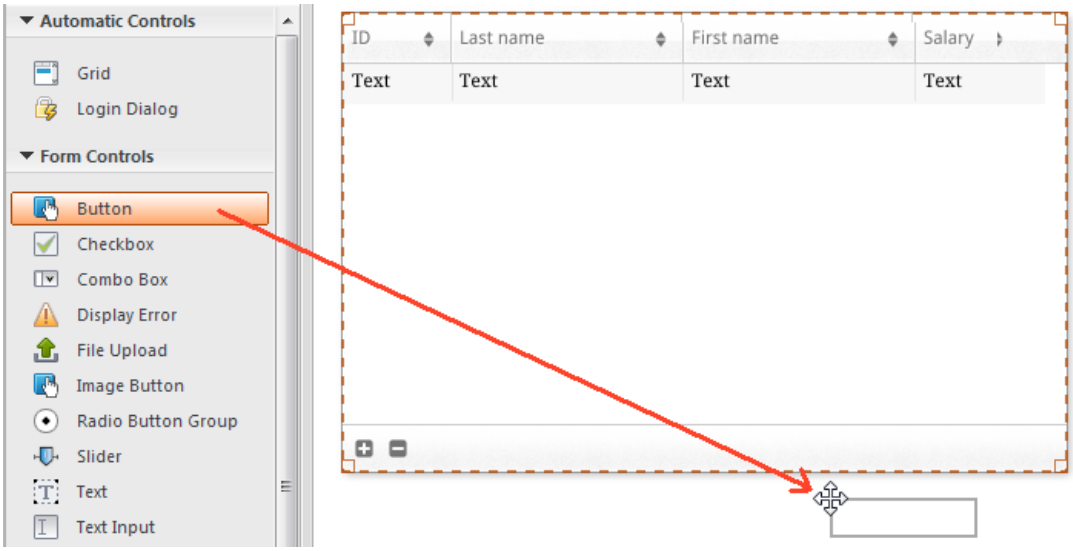
1. 4D 側で、DoubleSalaryという名前のプロジェクトメソッドを作成し、以下のコードを入力します:

```
FIRST RECORD([Employees])
While(Not(End selection([Employees])))
   [Employees]salary:=[Employees]salary*2
   SAVE RECORD([Employees])
   NEXT RECORD([Employees])
End while
```

2. メソッドプロパティのRESTの設定をし、**OK**をクリックします:

Wakanda では、クラスメソッドは以下のどれかに適用されます。エンティティ(レコード)、エンティティコレクション(セレクション)、データストアクラス(全レコード)。これらの内容を4D側で指定する必要があります。

3. Wakanda Enterprise Studio 側で、GUI Designer の中の **Index** に戻り、ウィジェットのリストからボタンを選択して追加します:



4. ボタンをダブルクリックし、"Double salaries"という名前をつけます:


5. "Double salaries"ボタンが選択されているのを確認したうえで、*GUI Designer* の右側にある**Events** ボタンをクリックします。
6. "On Click"のアイコンをクリックし、イベントを追加します:

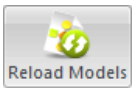

コードエディターが表示され、ボタンがクリックされたときに実行したいコードを記述することができます。ここでは単純に4Dの DoubleSalary メソッドを呼び出し、コールバックファンクション(*onSuccess*)にて全レコードをリロードするようにトリガーします。
7. 以下のコードを記述します:

```
sources.employees.DoubleSalary({
        onSuccess:function(event){
            sources.employees.allEntities();
        }});
```

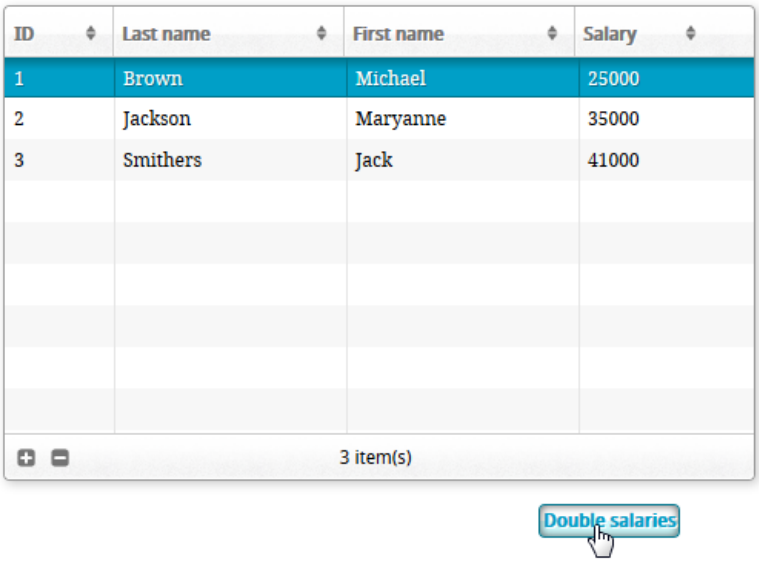
コードエディター内は以下の様になるはずです:

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

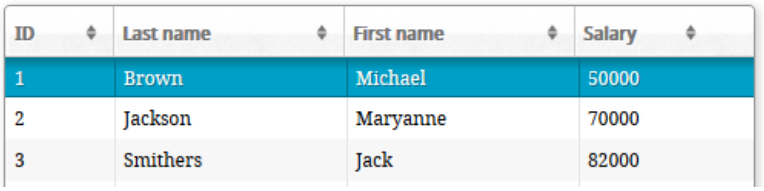
"employees"という単語は頭の"e"が小文字になっていることに注意して下さい。ここではクラスがウィジェットと関連付けられた際に自動的に作成されたデータストアクラスを使用しているからです。

8. エディターのツールバーの **Save**  ボタンをクリックして保存します。
これで4Dのメソッドを呼び出すテストの準備が出来ましたが、その前にモデルを Wakanda Enterprise Server上でリロードする必要があります。
9. Wakanda Enterprise Studio のツールバーの中にある**Reload Models** 

ボタンをクリックします。
10. ブラウザのページを再読み込みして **Double salaries** ボタンを表示させ、ボタンをクリックします。
Refresh your browser page so that the **Double salaries** button appears and then click on this button:



Salaryの欄の値が倍増したのが確認できます。
You can see that the salary values have all doubled:



ただし、ここで紹介した例はあくまでWakanda/4Dコネクターの設定を解説するためのものであり、ここで紹介した簡略化されたメソッドは製品では使用できるものではないことに注意して下さい。

4D データベースの設定

セキュリテイ上・パフォーマンス上の理由から、REST要求を使用しての4Dデータベースのテーブル、データ、そしてメソッドへの接続(Wakanda サーバーが使用するプロトコル)は、有効化され、明示的に認証されている必要があります。そのためには3段階のアクセスの設定をしなければなりません:

- Wakanda REST サービスのスタートアップ
- REST アクセスの管理(任意ですが推奨されます)
- それぞれのデータベースオブジェクト(テーブル、属性、プロジェクトメソッド)のRESTサーバーへの公開は必要に応じて個別に設定する必要があります。初期設定では:
  - テーブルと属性は全てRESTからアクセス可能
  - プロジェクトメソッドはRESTからアクセス不可

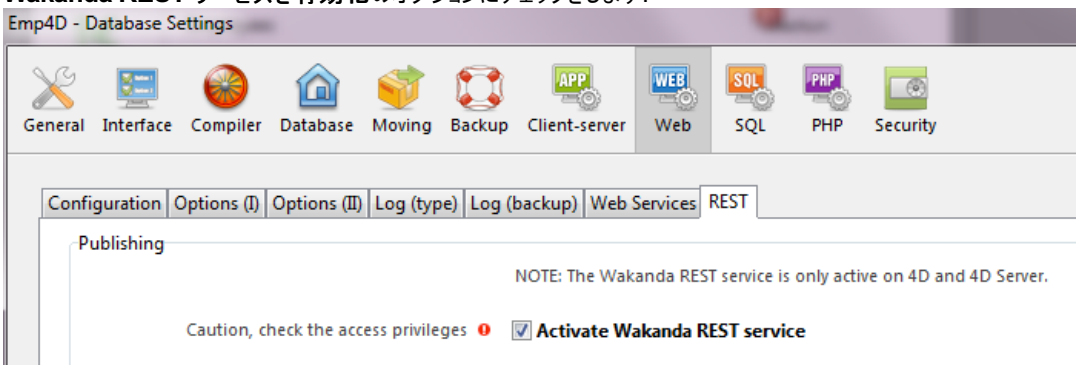
Wakanda RESTサービスの有効化

デフォルトとして、4D Server の v14 はREST 要求には反応しません。Wakanda/4Dコネクターの設定ができるようにこれらの要求が処理されるようにするためには、Wakanda RESTサービスを有効化する必要があります。

注: REST サービスは 4D HTTP サーバーを使用します。そのため、4D Webサーバーまた4D Serverが開始されていることを確認して下さい。

Wakanda REST サービスを有効化するためには以下の手順に従って下さい:

1. データベース設定においてWebのページのREST タグをクリックします。
2. **Wakanda REST サービスを有効化**のオプションにチェックをします:



Wakanda REST サービスが有効化されると、「警告:アクセス権が正しく設定されているか確認して下さい。」という警告メッセージが表示されます。これはREST接続が適切に管理されていない限り、デフォルトでデータベースオブジェクトへは自由にアクセスできてしまうためです(詳細は以下を参照して下さい)。

REST 接続の管理

REST 接続の管理とは、Wakanda RESTリクエストの後に、4D側でそのセッションを開くかどうかの認証をするということです。

Note: Wakandaを通したREST 接続の場合、mergeOutsideCatalog() メソッドの実行 の際に送られた名前とパスワードによって認証が行われます。

包括的にREST接続を管理する方法は二つあります:

- 4D パスワードを用いて自動的に管理する方法
- 新しい On REST Authentication データベースメソッドを用いてプログラムによって管理する方法

これらのコントロールモードはどちらかしか選択できません。つまり **On REST Authentication** データベースメソッドが定義されると4D パスワードを使用した自動アクセス管理は無効化されます。

警告:これらのアクセス管理がどちらも有効化されていない場合、RESTを経由してのデータベースへのアクセスは常に受理されてしまいます。この状態は推奨されません。

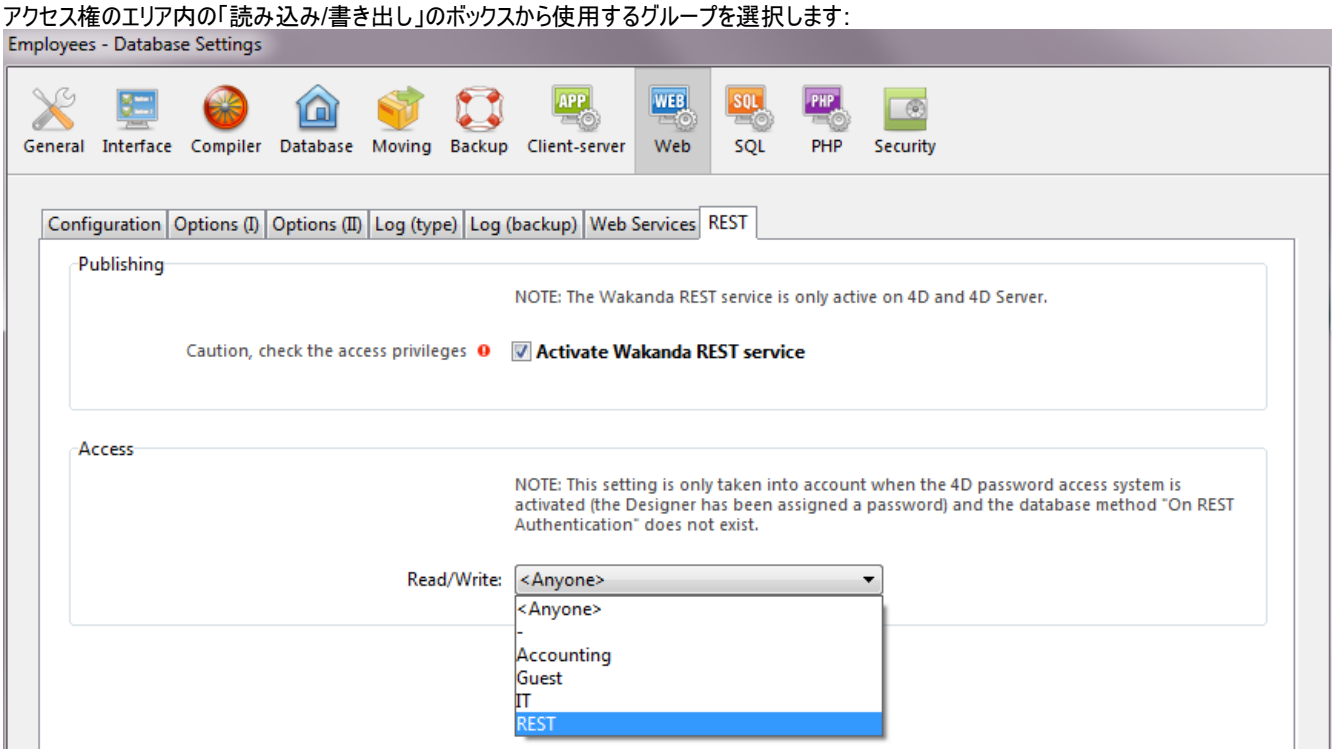
4D パスワード使用した自動コントロール

4D では、Wakanda アプリケーションから4D サーバーへのリンクを設定できる4Dユーザーのグループを指定することが出来ます。
以下の手順でアカウントを指定して下さい。

4D では、Wakanda アプリケーションから4D サーバーへのリンクを設定できる4Dユーザーのグループを指定することが出来ます。

以下の手順でアカウントを指定して下さい。:

1. データベース設定の画面からWeb→ RESTのページを表示。
2. アクセス権のエリア内の「読み込み/書き出し」のボックスから使用するグループを選択します:



初期設定では、メニューには**<すべて>**と表示されています。これはREST接続は全てのユーザーにオープンであるという状態を示しています。

グループの指定が終わると、そのグループに所属するユーザーのみがWakanda RESTリクエストを通して4Dへとアクセスできるようになります。具体的には、4D Server上で **mergeOustideCatalog()** メソッドを使用してセッションを開くことができるということです。このグループに所属していないアカウントの場合は、4Dはリクエストの送信者に対して認証エラーを返します。

この設定を有効にするために以下の点に注意して下さい:

- 4D パスワードシステムが起動している(パスワードがDesignerに割り当てられている)必要があります。
- **On REST Authentication** が定義されていないことを確認して下さい。定義されてしまうと、データベース設定のアクセス設定が全て無効となってしまうからです。

On Rest Authentication データベースメソッドを使用する方法

On REST Authentication データベースメソッドを使用することによりRESTセッションの接続の設定を自在に管理できるようになります。メソッドが定義されると、サーバーがRESTリクエストを受けた時に4D または4D Serverから自動的に呼び出されます。

Wakanda Serverから **mergeOutsideCatalog()** メソッドを使用したRESTセッションを開くリクエストが来ると(一般的なケース)、接続の識別子がリクエストのヘッダーに供給されます。続いて On REST Authentication データベースメソッドが呼ばれこれらの識別子を評価します。4D データベースのユーザーのリストを使用することもできますし、独自の識別子のテーブルを使用することもできます。

詳細な情報に関しては、4D *Language Reference* の On REST Authentication データベースメソッドの詳細を参照して下さい。

REST に公開されている 4D オブジェクトの設定

Wakanda REST サービスが4D データベース内で有効化されると、デフォルトでREST セッションは全てのテーブルとフィールドにアクセスすることができ、またそのデータを使用することが出来ます。例えば、あるデータベースに[Employee]というテーブルがあった場合、Wakanda側で以下の様に記述することでデータを取得することができます:

```
var emp=ds.Employee.query("name == 'Martin'");
    //名前のフィールドが`Martin`である従業員の全データを返します。
```

Note: 「非表示」のオプションにチェックがされている4D のテーブル/フィールドに関しても、RESTサーバーへと公開されます。

さらにWakandaサーバーは4D データベースのプロジェクトメソッドにアクセスることもできます。しかしながら、セキュリティ上の理由からこのアクセスはデフォルトでは無効化されています。

データベースのオブジェクトのRESTへの公開をカスタマイズしたい場合は:

- 公開したくないテーブル/フィールドは「RESTサーバーに公開」のチェックを外します。
- 公開したいテーブル/フィールドは「RESTサーバーに公開」にチェックをします。

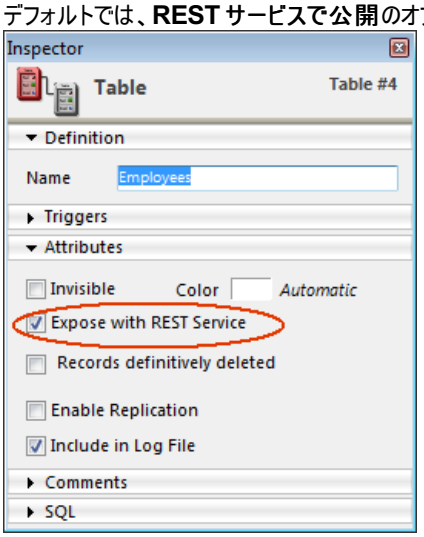
REST リクエストが認証されていないリソース(テーブルまたはプロジェクトメソッド)にアクセスをしようとした場合、4Dはエラーを返します。

テーブルの公開

デフォルトでは、全てのテーブルはRESTサーバーに公開されています。

セキュリティ上の理由からデータベースの一部のテーブルのみRESTサーバーに公開したいという場合があるかもしれません。しかし、ユーザー名とパスワードを記録した[Users]というテーブルを作成していた場合は、これは公開しない方がよいでしょう。

テーブルのRESTサーバーでの公開は以下の手順で修正します:

1. ストラクチャーエディター内で公開したいテーブルのインスペクターを表示します。
   デフォルトでは、**REST サービスで公開**のオプションにチェックがされています:


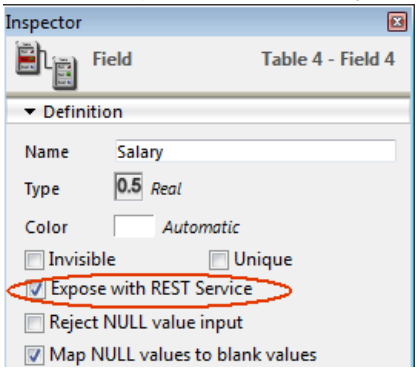
2. **REST サーバーで公開**のオプションのチェックを外します。
   または
   公開するテーブルに関してはオプションにチェックをしてください。
   公開・非公開を修正したいテーブルそれぞれに関して上記の操作をして下さい。

フィールドの公開

デフォルトでは、フィールドは全て REST サーバーで公開されています。

しかしテーブル内のフィールドのうち、REST サーバーで公開したくないものもあるでしょう。例えば、[Employees]というテーブルの給料のフィールドなどは公開したくないかもしれません。

フィールドごとのREST公開については以下の様に修正します:

1. ストラクチャーエディター内で公開したくないフィールドのインスペクターを表示します。
   デフォルトでは、**REST サービスで公開**のオプションにチェックがされています:


2. **REST サーバーで公開**のオプションのチェックを外します。
   または
   チェックされていないフィールドを公開するためにはチェックをします。
   公開・非公開を修正したいフィールドそれぞれに関して上記の操作をして下さい。

フィールドがRESTで公開すためには、テーブルも同様に公開されてなければならないことに注意して下さい。親のテーブルが公開されていないとき、その中のフィールドは公開状態に関係なく非公開になります。これを利用して、テーブルの REST での公開設定を選択することにより、個々のフィールドの**RESTサービスで公開**の設定を変えることなく公開/非公開を切り替えることが出来ます。

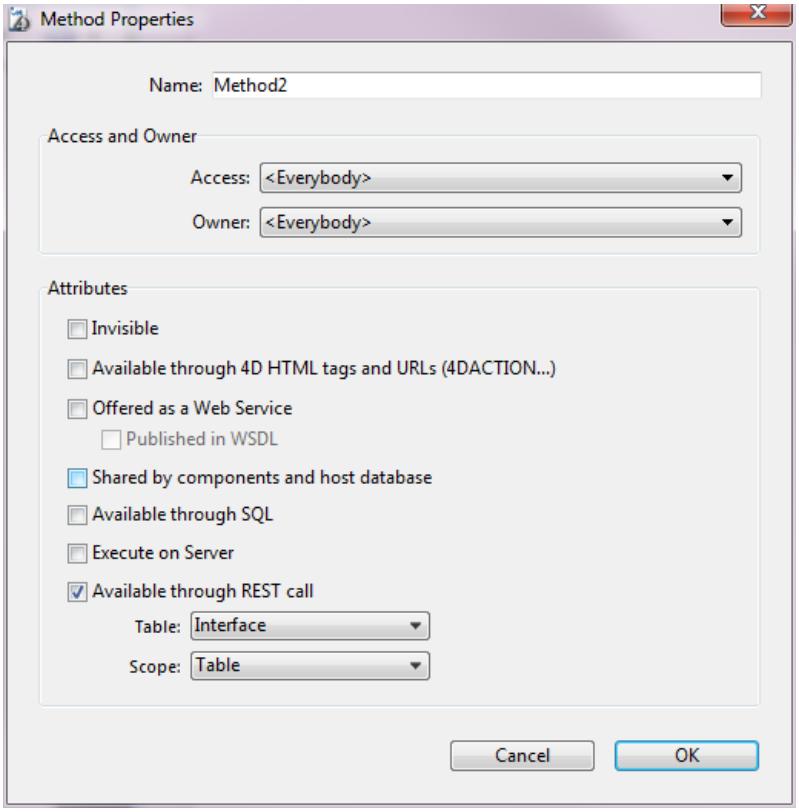
プロジェクトメソッドの公開

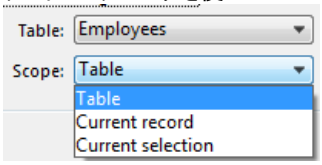
デフォルトではどのプロジェクトメソッドもRESTでは公開はされていません。

しかし、場合によっては一部のプロジェクトメソッドをRESTに公開したいことがあるかもしれません。そのためには適切なオプションを選択し、メソッドの実行コンテキストを定義する必要があります。

Note: 4D メソッドにアクセスグループが関連付けられている場合、RESTのグループがこのグループに含まれている必要があります。

プロジェクトメソッドのREST公開は以下の様に設定します:

1. メソッドプロパティのダイアログボックスを表示します。
   注: メソッドプロパティのダイアログボックスは、エクスプローラー内のメソッドのページのコンテキストメニューか、メソッドエディターの**メソッド情報**のボタンから行くことが出来ます。
2. **REST呼び出しからの利用を許可**のオプションにチェックを入れます:


3. テーブルとスコープを使ってREST実行コンテキストを定義します。


   これらの設定はRESTシンタックスとWakandaのロジックでは必須です。この点についての詳細は、以下のセクションを参照して下さい。
4. **OK**をクリックして変更を確定させます。
   RESTを介して使用可能なプロジェクトメソッドは、4Dエクスプローラーの「RESTメソッド」内に一覧で表示されます(以下の "エクスプローラー" の章を参照して下さい)。

プロジェクトメソッドのペアレントテーブルとスコープについて

RESTリクエストを介して使用可能なプロジェクトメソッドを宣言するとき、その呼び出しコンテキストを**テーブル**と**スコープ**を通じて明示的に宣言する必要があります:

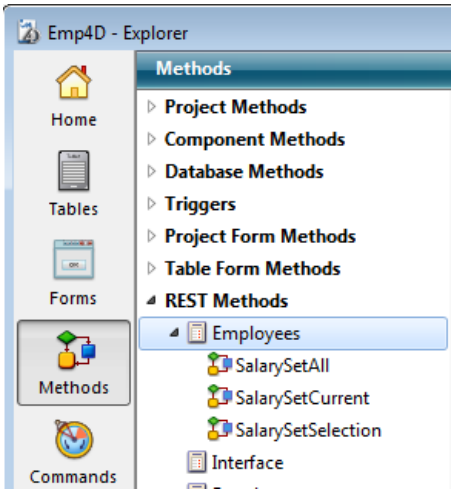
- **テーブル:** プロジェクトメソッドと関連付けられているテーブルです。REST呼び出しの最中、メソッドは */tableName/methodName* の形の構文で呼び出すことができます。
  メニューにはRESTに公開されているデータベースのテーブルの一覧が表示されます。メソッドが扱うデータを含むテーブルを選択して下さい。また、例えば[RESTInterface] というテーブルを作成してRESTに公開されている全てのプロジェクトメソッドを関連付けることもできます。

- **スコープ:** メソッドが適用される範囲を指定します。この宣言は必須です。なぜなら、Wakanda側ではメソッドはJavaScriptオブジェクトのプロパティとしてみなされ、これらのオブジェクトを使用しないと呼び出せないからです。公開されている4Dメソッドはそれぞれ明示的に呼び出されるデータベースコンテキストと関連付けられている必要があります。**テーブル**、**カレントレコード**、そして**カレントセレクション**から選択できます。
  - **テーブル:** このオプションは、4Dメソッドが指定されたテーブルの全てのレコードを使用して実行されるという事を意味します。Wakanda側では、メソッドは *Datastore class* という型を使用して呼び出されます。
  - **カレントレコード:** このオプションは、4Dメソッドが指定されたテーブルのカレントレコードを使用して実行されるという事を意味します。Wakanda 側では、メソッドは *Entity* という型のオブジェクト上で呼び出されます。
  - **カレントセレクション:** このオプションは4Dメソッドが指定されたテーブルのカレントセレクションを使用して実行されるという事を意味します。Wakanda 側では、メソッドは *Entity Collection* という型のオブジェクト上で呼び出されます。

警告: 4D側でプロジェクトメソッドの公開設定やスコープの設定を変更した場合、Wakanda側でリモートモデルをリロードしてこれらの変更を有効化する必要があります。

エクスプローラー

Wakanda RESTサービスが有効化されているとき、RESTに公開されているテーブルとそれに関連付けられているプロジェクトメソッドはの一覧は、4Dエクスプローラー内のRESTメソッドのセクションに表示されます:



On REST Authentication データベースメソッド

$1, $2, $3 -> On REST Authentication データベースメソッド -> 戻り値

引数	型	説明
$1	テキスト	⬅ユーザー名
$2	テキスト	⬅パスワード
$3	ブール	⬅True = ダイジェストモード False = ベーシックモード
戻り値	ブール	➲True = リクエスト承認 False = リクエスト拒否

説明

On REST Authentication データベースメソッド はREST セッションを開くのを管理するための方法を提供します。このデータベースメソッドは主にWakanda Serverと4D v14との接続を設定するときにそれをフィルタリングするのが主な目的です。

Wakanda Server から ***mergeOutsideCatalog()*** メソッドを使用して REST セッションを開くリクエストが来ると(一般的なケース)、接続の識別子がリクエストのヘッダーに供給されます。続いて **On REST Authentication データベースメソッド** データベースメソッドが呼ばれこれらの識別子を評価します。4D データベースのユーザーのリストを使用することもできますし、独自の識別子のテーブルを使用することもできます。

重要: On REST Authentication データベースメソッド が定義される(つまり中にコードが記述される)と、4D はREST リクエストの管理をそちらに全て一任します。このとき、データベース設定のWeb/RESTページ内の「読み込み/書き出し」メニューで設定した内容は、無視されます(*Design Reference* マニュアルを参照して下さい)。

このデータベースメソッドは二つのテキスト型の引数($1 と $2)と一つのブール型の引数($3)を4Dから受け取り、ブール型の引数 $0 を返します。これらの引数は以下の様に宣言されている必要があります。

```
  //On REST Authentication データベースメソッド
 C_TEXT($1;$2)
 C_BOOLEAN($0;$3)
 ... // メソッドのコード
```

$1 には接続に使用したユーザー名が入り、$2 にはパスワードが入ります。

リクエストに使われるモードにより、パスワード ($2) は標準テキストまたはハッシュ値で受け取る事が可能です。このモードは $3 引数によって指定され、適切に処理することができます:

- パスワードが標準テキスト(ベーシックモード)である場合、$3 には**False** が渡されます。
- パスワードがハッシュ値(ダイジェストモード)である場合、$3 には **True** が渡されます。

REST接続リクエストがWakanda Serverから来るときは、パスワードは必ずハッシュ値で送られてきます。

リクエストがブラウザや Wakanda 以外の Web クライアントから送られてくる場合、デベロッパが責任を持って "username-4D" フィールドと "password-4D" フィールドを HTTP ヘッダーに含めることによってオリジナルのHTML/JavaScript ページからの認証を管理して下さい。この場合、パスワードは4D REST サーバーに標準テキストで送られてなければなりません(サードパーティからの干渉のリスクを避けるためにSSLを使用して下さい)。

REST 接続の識別子は、データベースメソッド内でチェックしなければなりません。通常、ユーザー独自のテーブルを使用して名前とパスワードをチェックします。もし識別子が有効であるなら、$0 に**True** を渡します。すると、リクエストが受理されます。4Dはこのリクエストを実行して結果をJSON形式で返します。
それ以外の場合は $0 に **False** を渡します。この場合、接続は拒否され、サーバーはリクエストの送信者へ認証をエラーを返します。

ユーザーがデータベースの4Dユーザーのリストの中に載っているとき、以下のコードによってパスワードを直接チェックすることができます:

```
 $0:=Validate password($1;$2;$3)
```

Validate password コマンドは拡張され、第一引数にユーザー名、第二引数にパスワードを渡し、任意の第三引数でパスワードがハッシュド形式で書かれているかどうかを指定できるようになりました。

4D データベースのものとは別の独自のユーザーリストを使用したい場合、そのユーザー達のパスワードを、Wakanda Server が **On REST Authentication データベースメソッド** データベースメソッドに接続リクエストを送る時のアルゴリズムと同じものを用いてハッシュ形式にて$2 引数に保存することができます。

この方法を使用してパスワードをハッシュする場合、以下の様に記述して下さい:

```
 $HashedPasswd :=Generate digest($ClearPasswd ;4D digest)
```

Generate digest コマンドにはハッシュアルゴリズムとして 4D digest を受け取れるようになりました。これは4Dのパスワードの内部管理で使用されているメソッドと対応しています。

例題 1

この例題ではパスワード "123"を使用する、4Dユーザーと合致しない "admin"というユーザーのみを受け入れる場合を考えます:

```
  //On REST Authentication database method
 C_TEXT($1;$2)
 C_BOOLEAN($0;$3)
  //$1: ユーザー
  //$2: パスワード
  //$3: ダイジェストモード
 If($1="admin")
    If($3)
       $0:=($2=Generate digest("123";4D digest))
    Else
       $0:=($2="123")
    End if
 Else
    $0:=False
 End if
```

例題 2

以下の **On REST Authentication データベースメソッド** の使用例は、接続リクエストが4D データベースのユーザーに保存されている二つの認証済みの Wakanda サーバーのどちらかから来ていることをチェックします:

```
 C_TEXT($1;$2)
 C_BOOLEAN($0)
 ON ERR CALL("REST_error")
 If($1="WAK1")|($1="WAK2")
    $0:=Validate password($1;$2;$3)
 Else
    $0:=False
End case
```

Wakandaアプリケーション側の設定

Wakanda Enterprise側では、通常 mergeOutsideCatalog() JavaScriptメソッドを使用して4D v14データベースと接続します。接続がWakandaと4Dの間に設立されると、Wakandaは4Dで公開されている全てのテーブル、属性、そしてプロジェクトメソッドをローカルオブジェクトと同じように使用することが出来るようになります。

またこのメソッドを使用して追加でJavaScriptコードを実行する事もできます。例えば、リモート属性のプロパティをローカルに修正したり、クラスをクラスを拡張したり、計算属性を追加したりできます。

mergeOutsideCatalog() メソッドの実行

mergeOutsideCatalog() JavaScript メソッドはリモートデータのカタログを指定し、それをカレントの Wakanda モデル内で使用します。このメソッドはカレントのモデルに関連付けられている .js ファイル内で呼び出され、Wakandaサーバーによって実行されなければなりません。

このとき、以下の二つのシンタックスのどちらかを使用できます:

- ダイレクトシンタックス:

```
model.mergeOutsideCatalog(localName, address, user, password);
```

- オブジェクトを使用したシンタックス:

```
model.mergeOutsideCatalog(localName, {
    hostname: address,
    user: userName,
    password: password,
    jsFile: jsFilePath
    timeout: minutes });
```

オブジェクトを使用したシンタックスの利点は、4Dデータベースに接続したあとに実行される.jsファイルを追加できることです。このファイルはリモートデータベースから参照されるカタログをローカルに修正することができます。

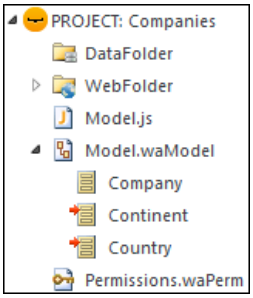
引数	型	説明
localName	文字列	リモートカタログのローカル名
ipAddress	文字列	リモートデータサーバーのアドレス(セキュリティのためにHTTPSを使用して下さい)
userName	文字列	セッションを開くためのユーザー名
password	文字列	セッションを開くためのパスワード
jsFilePath	文字列	JavaScriptファイルへのパス名(任意)
timeout	数字	4Dデータベースへのクライアント接続のタイムアウト(分、初期設定は60。任意)

より詳細な情報に関しては、Wakanda Server-Side API manual の、documentation of the mergeOutsideCatalog() メソッドを参照して下さい。

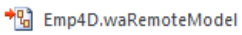
model

model オブジェクトは、Wakandaアプリケーションのカレントの「モデル」をあらわします。つまり、Wakandaの"datastore classes"(テーブル)とメソッド一式のことです。4D Mobileアーキテクチャにおいては、Wakanda モデルは空であっても構いません。Wakanda アプリケーションにオブジェクトが含まれる場合、リモート4Dアプリケーションから参照されたクラスとメソッドはローカルのモデルと組み込みされます。

接続が正常に確立されると、「公開」されている4D テーブルがWakanda側のモデルのクラスのリストに表示されます。Wakanda Enterprise Studio側では、リモートテーブルがローカルモデルのクラスの一覧の中に表示されます。外部要素は赤い矢印で表示されます:

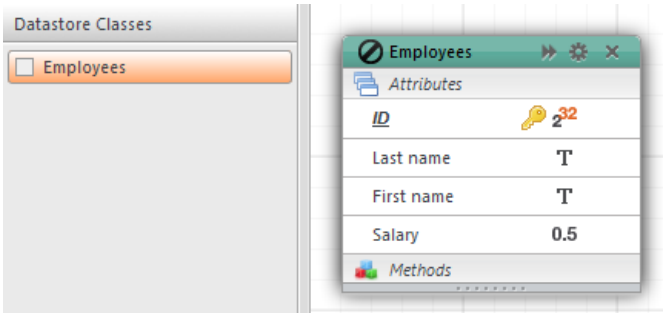


外部カタログはWakanda Studio側でもlocalName.waRemoteCatalog というファイル名で表示されます:



Note: ファイルの拡張子はWakanda Studioでは隠すことができます。

このファイルをダブルクリックすることにより外部カタログをWakanda Enterprise Studio内で見ることができます:



jsFile

jsFile プロパティにはモデルと同じフォルダ内にあるJavaScriptファイルへの相対パスを渡すことが出来ます。このファイルは外部カタログが組み込まれた後、Wakanda によって実行されます。また、このファイルはカスタマイズ、最適化、セキュリテイなどの目的ためにモデルのローカルバージョンを変更することができます。さらに、このファイルを使用することによって以下の様なことが可能です:

- イベントやスコープなどのデータスコアクラス属性のプロパティを変更できます。以下の様に記述します:

  ```
  model.className.attributeName.scope ="publicOnServer"
  ```

- データスコアクラスに計算属性を追加することができます。以下の様に記述します:

  ```
  model.className.attributeName.onGet = function()
  model.className.attributeName.onSet = function()
  ```

- 外部カタログのテーブルから派生させたローカルのデータストアクラスを作成し、クライアントへ送られるデータを完全に管理することができます。派生されたデータストアクラスは外部テーブルのカスタムビューを表示することができる一方、Wakanda Server上の拡張された(親の) データストアクラスへもアクセスることができます。以下の様に記述します:

  ```
  model.DerivedClass = new DataClass("Emps", "public", "My4DTable")
  ```

- セキュリティのため、またはネットワークトラフィックを最適化するために、派生したデータストアクラスから属性を除去することができます。以下の様に記述します:

  ```
  model.DerivedClass = new DataClass("Emps", "public", "My4DTable")
  model.DerivedClass.removeAttribute("salary");
  model.DerivedClass.removeAttribute("comments");
  model.DerivedClass.removeAttribute("...");
  ```

  上記のコードは、"My4DTable"をもとに派生した"DerivedClass"という名前のクラスを作成し、このクラスはネットワークを使用して必要な属性のみを送ります。

モデルと組み合わせて使用できるJavaScript コードに関しては、Wakandaのドキュメントの中の Model API の章を参照して下さい。

例題

- ダイレクト接続の例:

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

- オブジェクトを使用した接続例:

  ```
  model.mergeOutsideCatalog("base4D", {
      hostname: "http://localhost:8050",
      user: "wak",
      password: "123456",
      jsFile: "Model2.js"
      timeout: 15 });
  ```

openRemoteStore() と addRemoteStore()

Wakanda と 4D 間の動的なリンクは、openRemoteStore() と addRemoteStore() を使うことによっても設定することができます。

これらのメソッドは **mergeOutsideCatalog()** のように、4D データベースのデータへのダイナミックなアクセスを可能にしますが、仕組みが異なります:

- これら二つのメソッドはWakanda セッション中であればソリューションがロードされたときでなくてもいつでもリモートモデルを参照することができます。
- 外部モデルのテーブル、属性、メソッドは、個別のデータストアを使用してアクセス可能です。Wakandaアプリケーションのローカルモデル(**ds** オブジェクトによってアクセス可)と統合はされません。

openRemoteStore() はカレントのJavaScriptのコンテキストの中でのみ有効な参照を返しますが、openRemoteStore() はセッションの間はずっと参照を維持し続けます。

より詳細な情報に関しては、Wakanda documentationの **openRemoteStore()** と**addRemoteStore()** についての説明を参照してください。

4Dテーブルとメソッドの呼び出し

4Dテーブルの呼び出し

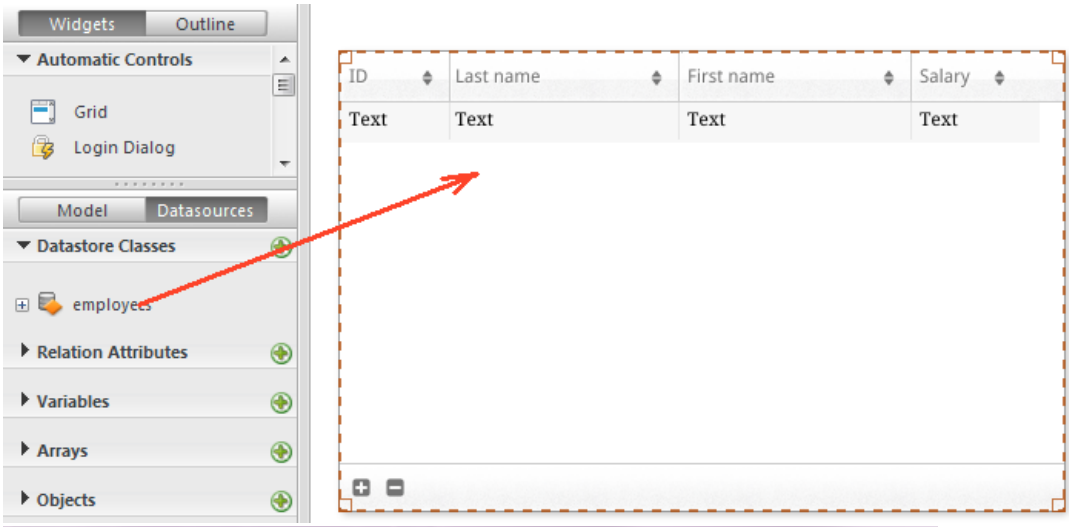
Wakanda アプリケーションから参照されている 4D テーブルは、データストアクラスと同じように、**ds** オブジェクトのプロパティとしてJavaScriptのコードの中で直接使用することができます。

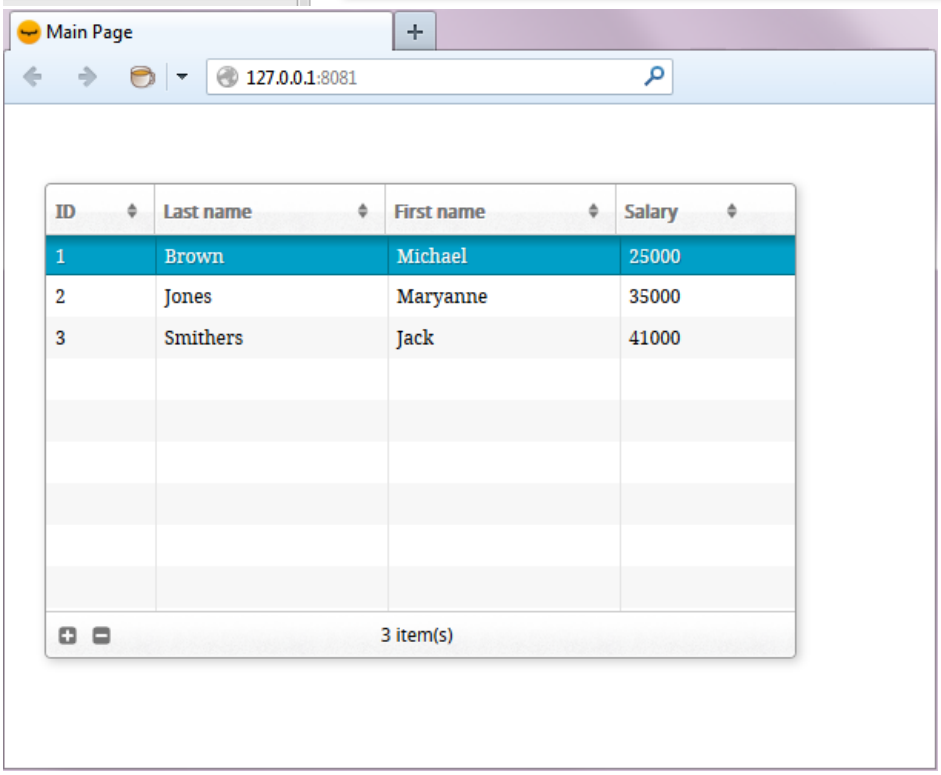
注: ds オブジェクトはWakandaのカレントのデータストアを内包しています。

例えば、[Employees]テーブルのレコード内でクエリを実行しようとした場合、以下の様に記述します:

```
var emp = ds.Employees.query("age > :1",30);
        // Employeesテーブルから、年齢が30 歳を超える
        // レコードのコレクションをemp変数に返します。
```

また、クライアント側では、ウィジェット付随の、データストアクラスに基づいた *datasources* 自動メカニズムを使用することもできます。例えば、`employees`データソースを`Grid`型のウィジェットと関連付けするとemployeesのリストが自動的に表示されます:





テーブルが*datasource*と関連付けられているとき、データソースを使用してテーブルのデータにアクセスすることもできます。例えば、`employees` データソースのレコードのコレクションをソートしたい場合、以下の様に記述します:

```
sources.employees.orderBy("age");
        //employees のコレクションを年齢順にソートします。
```

datasource クラスの仕様については、Wakanda documentationを参照して下さい。

4D メソッドの呼び出し

スコープとオブジェクト

Wakanda 内で参照されている4Dメソッドは、**datastore class**, **entity collection** または **entity** オブジェクトのプロパティとしてJavaScriptのコードの中で直接使用することができます。どれのプロパティとして呼び出されるかは4D側で定義されたスコープによって決まります(プロジェクトメソッドのペアレントテーブルとスコープについて を参照して下さい)。Wakandaオブジェクトとプロジェクトメソッドの対応表は以下の様になります:

4D スコープ	**Wakanda オブジェクト**
テーブル	datastore class
カレントセレクション	entity collection
カレントレコード	entity

Note: 4Dメソッドはデータソースを使用することによってクライアント側で呼び出すことも可能です(以下を参照して下さい)。この場合は全てのメソッドが使用可能で、データソースが状況に応じてカレントコレクションかカレントエンティティに適用するかを自動的に判別します。

例えば、前章で使用したクエリメソッドを使用してクエリを実行した場合、Wakandaはエンティティコレクションを返します。このコレクションに対しては、スコープが「カレントセレクション」と宣言されている4Dプロジェクトメソッドであればどれでも使用可能です。

サーバーとクライアント

4DメソッドがJavaScriptから呼び出される方法は3通りあります:

- SSJS Datastore APIを使用して(SSJS)サーバー上でJavaScriptを実行して呼び出し:この場合、4Dメソッドは先に説明のあったように **datastore class**, **entity collection** または **entity オブジェクト** のプロパティとして呼び出されます。以下の様に記述します:

  ```
  var vTot = ds.Emp.raiseSalary(param))
  ```

- Wakanda Ajax Framework (WAF)を使用して、クライアント上(ブラウザなど)で実行されたJavaScriptコードでJavaScriptから呼び出し:この場合、使用するAPIによって二通りの方法があります:
  - WAF Datasource API を使用する方法:このハイレベルなAPIはデータを管理するための様々な自動機能を提供します。このAPIを使用した場合、**datastore classes に関連付けられたデータソースのプロパティ**として呼び出され、内容に応じて自動的にデータストアークラス、カレントエンティティコレクション、もしくはカレントエンティティに適用されます。メソッドの戻り値やエラーを処理するのであれば、全て非同期シンタックスを使用して管理しなければなりません(クライアントでコードを実行するためには必須です)。記述例としては以下の様になります:

    ```
    sources.employee.raiseSalary(param,
        {onSuccess: function(event)
        { ... //メソッド終了時に実行されるべきコード}
    }))
    ```

    ここではコールバック関数の使用は必須ではありません。何故ならデータソースオブジェクトはクエリ後のカレントコレクションに合わせて表示を更新するなどの動作をサポートする自動機能があるからです。
  - WAF Dataprovider API を使用する方法:このローレベルなクライアントAPIを使用するとオブジェクトを直接扱うことができます。SSJS Datastore API 同様、4Dメソッドは **datastore class**, **entity collection**または **entity オブジェクト**のプロパティとして呼び出されます。しかしながらメソッドの戻り値やどのエラーも、非同期シンタックスを使用して管理しなければなりません(クライアントで実行されるコードのためには必須です)。記述例としては以下の様になります:

    ```
    ds.Employee.raiseSalary(param, // シンタックスはSSJSの呼び出しに
        {onSuccess: function(event)     // ていますが、これはクライアント側の
        // コードなので非同期呼び出しのコールバック関数を管理する必要があります。
        { ... //メソッド終了時に実行されるべきコード}
    }))
    ```

    呼び出す場所(サーバーかクライアント)と、使用すべきAPIはアプリケーションによって異なり、その詳細はWakanda ドキュメントに説明があります。

引数

標準的なメソッド同様、呼び出し中にメソッドに引数を渡す事ができます。これらの引数は$1、$2、、、という順番で引き受けられていきます。同じように、$0がメソッドからの返り値になります。

例題:給料が1500未満の従業員に対して5%の昇給を行いたい、という場合を考えます。

- 4D側では、IncreaseSalary メソッドをRESTサーバーに公開し、スコープを「カレントセレクション」に設定して、コードを以下の様に記述します:

  ```
  C_REAL($1)
  READ WRITE([Employees])
  FIRST RECORD([Employees])
  While(Not(End selection([Employees])))
     [Employees]salary:=[Employees]salary*$1
  ```

```
    SAVE RECORD([Employees])
    NEXT RECORD([Employees])
 End while
 UNLOAD RECORD([Employees])
```

- Wakanda 側では、以下のコードをサーバー上で実行します:

```
var emp = ds.Employees.query("salary < :1",1500);
        // emp にはsalaryが1500未満の従業員のコレクションが入ります。
    emp.IncreaseSalary(1.05);
        //コレクションに対してIncreaseSalary を実行します。
        //以下の様に記述することもできます:
        //"ds.Employees.query("salary < :1",1500).IncreaseSalary(1.05);
```

4Dコンテキストの更新

Wakanda リンクを通して4Dメソッドを呼び出す場合:

- メソッドがセレクション(*entity collection*)に対して適用されるとき、メソッドはカレントセレクションとなり、4Dはリンクをロードしたり有効化したりすることなくこのセレクションの最初のレコードに位置します。セレクションが空の場合、Selected record number コマンドは1ではなく0を返します。
- メソッドはレコード(*entity*)に対して適用されるとき、メソッドはカレントレコードとなり、読み書き可能な状態でロードされます。カレントセレクションはこのレコードのみに縮小され、Selected record number コマンドは1を返します。
- メソッドがテーブル(*datastore class*)に対して適用されるとき、カレントセレクションもカレントレコードも、どちら何も変更されません。

メソッドをRESTを通して実行した後、4Dのコンテキストは以下の様にリセットされることに注意して下さい:

- セレクションは0に減らされます。
- レコードはスタックが解除され、アンロードされます。
- プロセスにおけるローカルなセレクションとセットは破壊されます。
- メソッド実行中に開かれたトランザクションは全てキャンセルされます。
- フィールド、クエリデスティネーションまたはサーバー上のクエリの自動リレーションは全てリセットされます。
- 印刷ジョブはキャンセルされます。
- ウィンドウは閉じられます。
- SQL、PHP、またはHTTP 接続も閉じられます。

スコープエラー

4Dメソッドのスコープは、それを呼び出すWakandaオブジェクトの方と対応し、合致している必要があります。そうでない場合には *"TypeError: 'undefined' is not a function"* というエラーがWakandaによって返されます。

例えば、以下のコードによって記述された"getcursel"という4Dメソッドについて考えてみましょう:

```
 $0:=Records in selection([Table_1])
```

Wakanda側に以下のメソッドが実行されていると仮定します。:

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

query() メソッドはコレクションを返します。もし *getcursel* メソッドのスコープが「カレントレコード」に設定されていた場合、Wakandaは以下のエラーを返します:
TypeError: 'undefined' is not a function (evaluating 'ds.Table_1.query("Field_2 = 'a'").getcursel()')".*

リレーションの使用

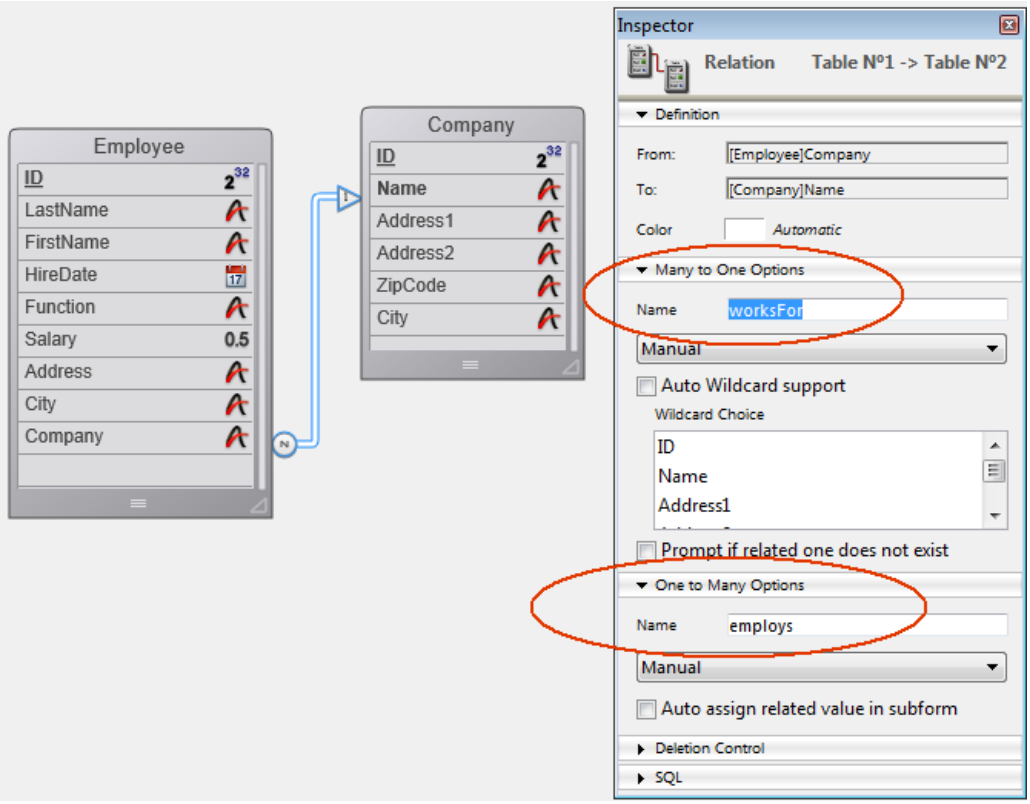
4Dテーブルとの間に設定されたリレーションは4D Mobileリンクのコンテキストにおいて透過的に使用されます。しかしながら、これらのリレーションのWakandaでの表示のされ方はモデルレベルで異なります。モデルエディターでは、リレーションはrelational attributesと呼ばれる特定の属性とリンクされています。これらの属性はリンクしたデータを直接表示するため、もしくはクエリを実行するためなどに使用されます。これについての詳細な情報に関しては、Wakandaドキュメントの"Attributes"セクションを参照して下さい。

4D側で設定されたそれぞれのリレーションに対して、Wakanda側ではモデルの表示に二つのrelational attributesが追加されています:

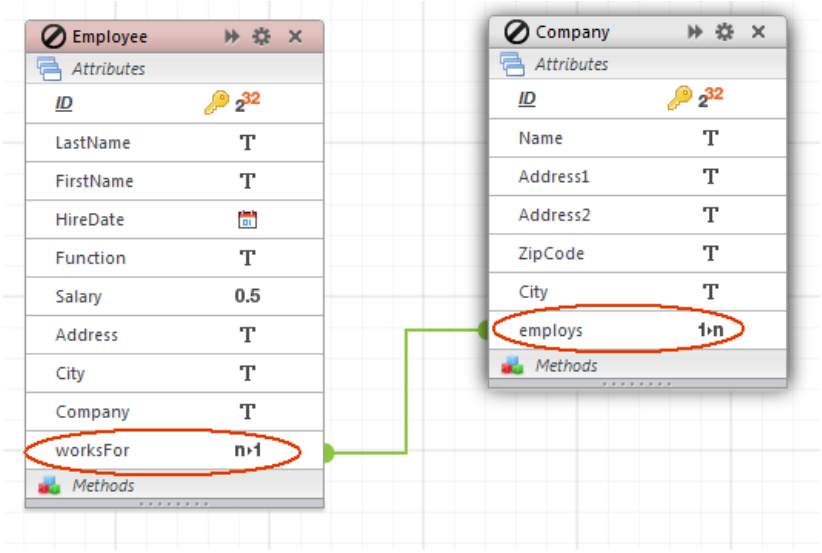
- リレーションのソーステーブル(クラス)内にて、n->1属性
- リレーションのデスティネーションテーブル(クラス)にて、1->n属性

これらの属性は、どちらも4D側のインスペクターにて定義されたN対1オプションと1対Nオプションでのリレーションの名前が与えられます。

具体例を考えましょう。"Employee/Company"ストラクチャーでのコンテキストにおいて、[Employee] テーブルから [Company] テーブルへのリレーションを作成したとします。このリレーションには、識別のために名前を付けることが出来ます。この場合、例えばN対1リレーションには"*worksFor*"という名前をつけて、1対Nリレーションには"*employs*"という名前を付けることが出来ます:



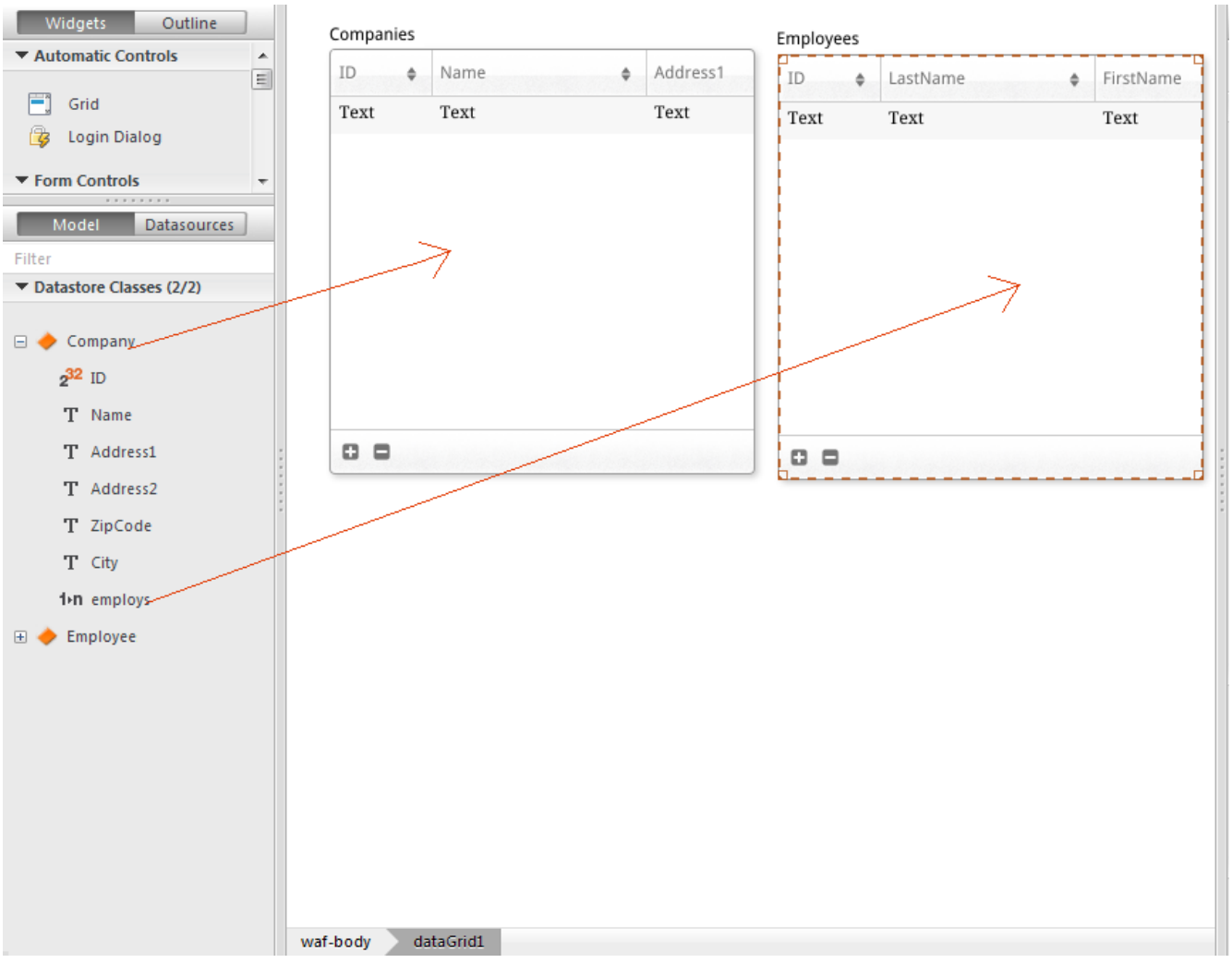
Wakanda側では、コネクターのリンクを通じてこれらのリレーションが二つの新しいrelational attributesによって自動的にマテリアライズされます。これはモデルエディター内にて確認することが出来ます:



これらのリレーション(ひいてはそれに対応するrelational attributes)には、アプリケーションの目的に応じて自由に名前をつけることができます。

これの利点は、Wakanda 側でこれらの属性を使用してリレートしたデータを扱うのが簡単になるという事です。具体的には、relational attributesに基づいて *datasources* と関連付けられたウィジェットを作成することが出来ます。これらのウィジェットはユーザーアクションに応じて自動的に管理・更新されます。

例えば、一つのグリッドに会社の一覧を、もう一つのグリッドに選択した会社の従業員を表示するようなページを簡単に作成できるようになります。"Company" datastore classを一つのグリッドに関連付け、"employs" relational attributeをもう一つのグリッドに関連付けるだけです:



対応するデータソースは自動的に作成され、実行の間、両グリッドは自動的に同期されます:

Companies

ID	Name	Address1
1	Gizmo Computers	12332 Madis
2	Pepperson Pipes	2293 Park St
3	Solstice Systems	2332 Market
4	Carmelito Cosmetics	2369 Rodeo
5	Seaside Candies	93 Amsterda

5 item(s)

Employees

ID	LastName	FirstName
1	Parker	John
2	Jameson	Henry
3	Johnson	Susan
4	Clarkson	Claire
5	Marker	Carl

5 item(s)

4D Mobileのセキュリティについて

4D データベースのテーブルから REST を通じて公開されたデータが Wakanda カタログと統合されたあとは、一部のデリケートなリソースに関してはアクセスを制限する必要があります。

4D とは違い、Webアプリケーションではインターフェースを使用して公開されているデータを管理することはできません。例えば、あるフィールドが表示されて いないからといって、それがユーザーからアクセスできないわけではない、ということです。HTTPリクエストとJavaScriptを使用することで、悪 意あるユーザーがプロテクトが不完全な Webサーバーから自由にデータを取得してしまう事態も起こり得ます。

この章では4D Mobileアプリケーションにおいてセキュリティ面で取るべき全ての対策を挙げているわけではないですが、公開しているデータを保護するために最低限必要な情報がまとめられています。

- **4D データベースへの REST アクセスの保護:** REST接続のリクエストは保護されている必要があります。以下二つのどちらかを使用しましょう:
  - 4D パスワード(4D パスワード使用した自動コントロール を参照のこと)
  - On REST Authentication データベースメソッド データベースメソッド
- **4D 側で REST サーバーへの公開を管理**: RESTサーバーへの公開・非公開はそれぞれのテーブル、属性、そしてメソッドごとに設定することができます。本当に必要なデータとメソッドのみ公開するようにしましょう。例えば、使用していないフィールド等は公開する必要はありません。
- **公開されているデータの保護:** ブラウザ経由で公開されているデータに関しては、Wakandaのセキュリティシステムを使って管理して下さい。以下の様にいくつかの手段があります(同時に複数併用することも可能です):
  - **スコープの調整**:Wakandaにて、モデルレベルで4Dデータベースとメソッドの属性のスコープの調整をします(Wakandaのドキュメント内のfor attributes または for methods の **scope** のプロパティを参照して下さい)。特に、スコープを **Public on Server** に設定するとサーバーからはコード実行のために自由にアクセスできますが、Webクライアントからはアクセスできなくなります。
    この設定は **mergeOutsideCatalog()** メソッドによって実行される追加の.js ファイルによって設定することができます。詳細な情報に関してはWakanda ドキュメントの Model API のページを参照して下さい。
  - **計算属性を使用：**計算属性は標準の属性と同じように使用できますが、その値には特定の関数 (**onGet、 onSet** 等) を通してのみアクセスできます。これはつまり、4D データベースのフィールドを直接公開せずに必要な計算属性のみを公開するといったことができるということです。4D フィールドへのアクセスは Wakanda サーバーから安全な方法で実行されます。
    動的な属性は、**mergeOutsideCatalog()** メソッドによって実行される追加の.js ファイルへと追加することができます。詳細な情報に関してはWakanda ドキュメントの Attributes を参照して下さい。