

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](#)
- [ステップバイステップ形式での解説](#)
- [4D データベースの設定](#)
- [Wakandaアプリケーション側の設定](#)
- [4Dテーブルとメソッドの呼び出し](#)
- [リレーションの使用](#)
- [4D Mobileセッションの管理](#)
- [4D Mobileのセキュリティについて](#)

4D Mobile アーキテクチャー

システム要件

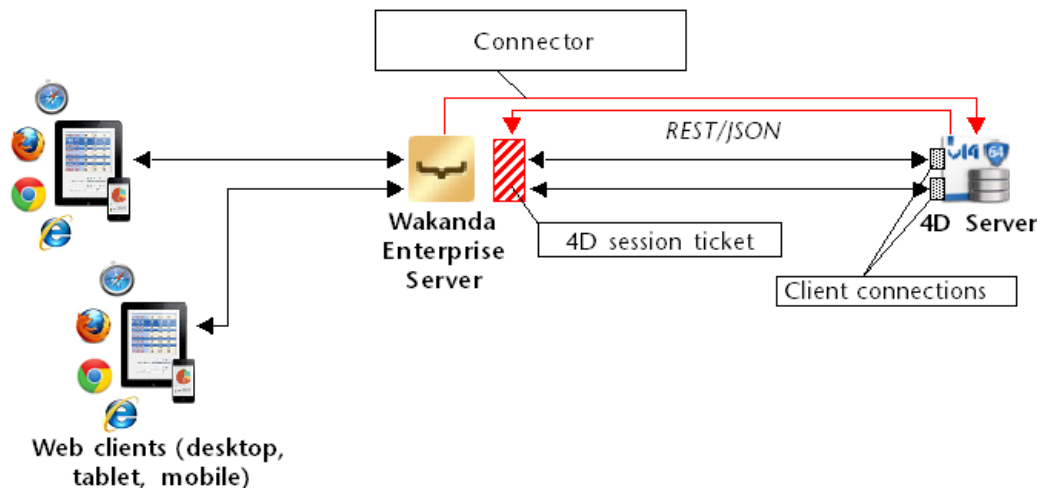
4D/Wakandaコネクタを使用するアーキテクチャーを設定する場合、以下のシステム要件が最低限必要になります：

- **4D シングルクーザー**(プロフェッショナル版、4D Mobile コネクタを使用してソリューションを開発・テストするために必要になります。この場合、4D Mobile接続は最大で同時に3つまでご利用になれます)、または、4D Mobile エクステンションパックを付けた **4D Server** (この場合4D Mobile接続は最大で同時に2つまで可能です)。
- **Wakanda Enterprise Server**と **Wakanda Enterprise Studio**
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できます：



Wakanda ソリューションが開始されると、Wakanda Enterprise serverは "Connect to Remote Datastore"ダイアログボックスまたはJavaScript接続メソッドにて定義された設定に従って4D Serverとのリンクを構築します。この接続が4D Server側で受理される([REST 接続の管理](#)の章を参照して下さい)と、4D Mobileセッション"チケット"がWakanda Serverへと配布されます。このチケットはWakandaによって、これ以降の全てのリクエストに対して使用されます。

このリンクを使用して、Wakanda server は4Dデータベース内の二つのタイプのリソースにアクセスすることが可能になります：

- テーブルとその属性(そのデータも含まれます)
- プロジェクトメソッド

接続が認証されると、これらのリソースはあたかもWakanda のローカルカタログに所属してたかのように、Wakanda 側から直接使用されます(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4D Mobile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最初に接続するときに引数によって変更することが可能です。セッション中に、4Dサーバーにて認証された4D Mobile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。

ステップバイステップ形式での解説

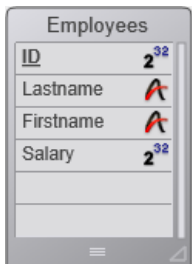
このセクションでは、ステップバイステップ形式で一つずつ手順を追って Wakanda / 4D コネクタの機能を紹介していきます。具体的には以下のような手順を解説します：

- 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)

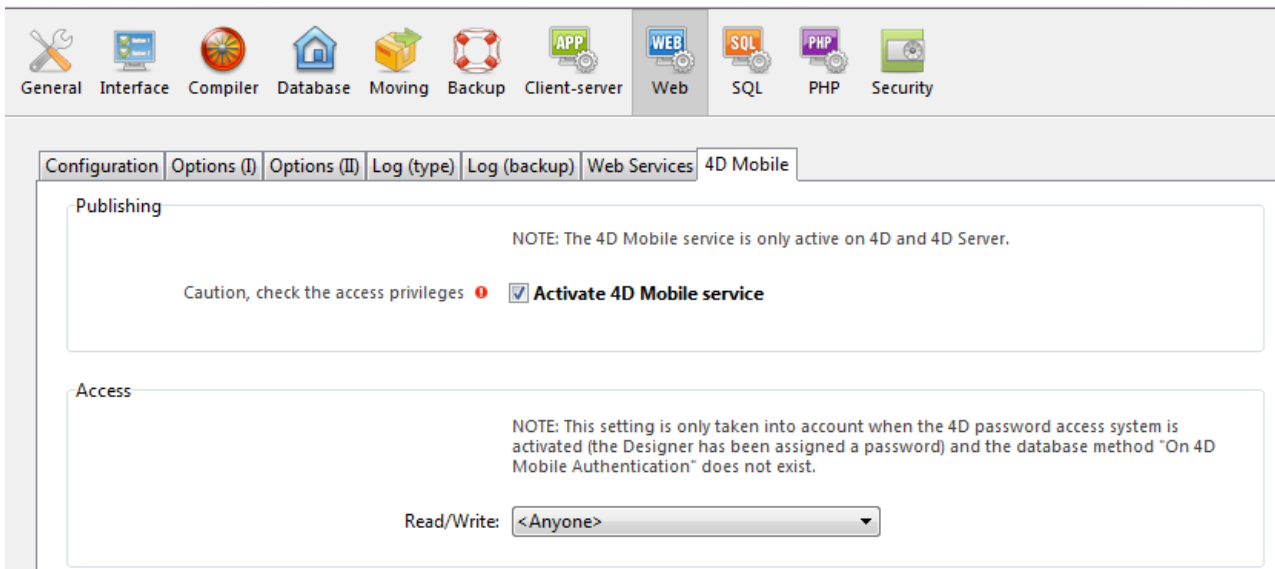


"4D Mobileサービスで公開"の属性は、全てのテーブルにおいて最初からチェックがされており、この設定は変更しないで下さい。

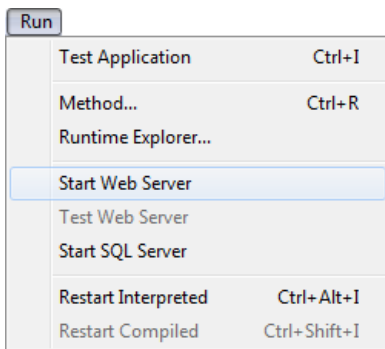
3. **Tables** のボタンをクリックして4Dにデフォルトフォームを作成させたのち、実際のデータを数レコード分作成します：

ID :	Lastname :	Firstname :	Salary :
1	Brown	Michael	25000
2	Jones	Maryanne	35000
3	Smithers	Jack	41000

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、"**4D Mobile**"タブをクリックします。
5. "4D Mobile 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：



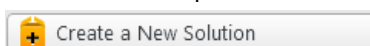
6. 実行のメニューの中から**Web サーバー開始**を選択します:



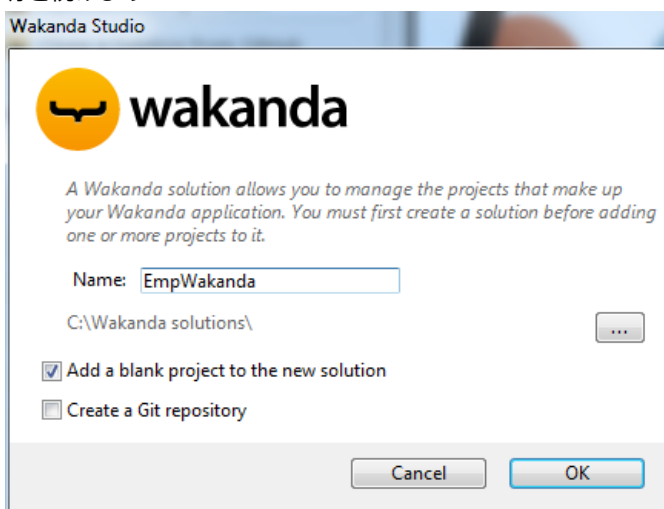
これで4DデータベースはWakanda からの4D Mobile リクエストに応答する準備が出来ました。なお、ここでは簡略化のために4D Mobile 接続の管理まではしていないという点に注意して下さい。実際の製品やオープンアーキテクチャの場合は4D Mobile 接続を安全に管理することが必要不可欠となります(詳細な情報に関しては[4D Mobileのセキュリティについて](#) を参照して下さい)。

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:

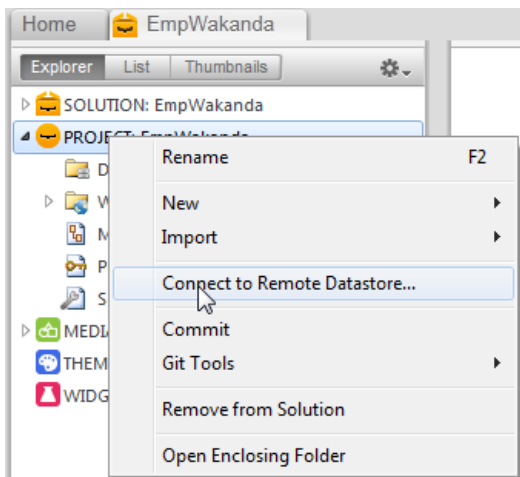


2. 作成ダイアログボックスにて、名前を記入して **OK**をクリックします。ここでは"EmpWakanda"という名前をつけて説明を続けます:

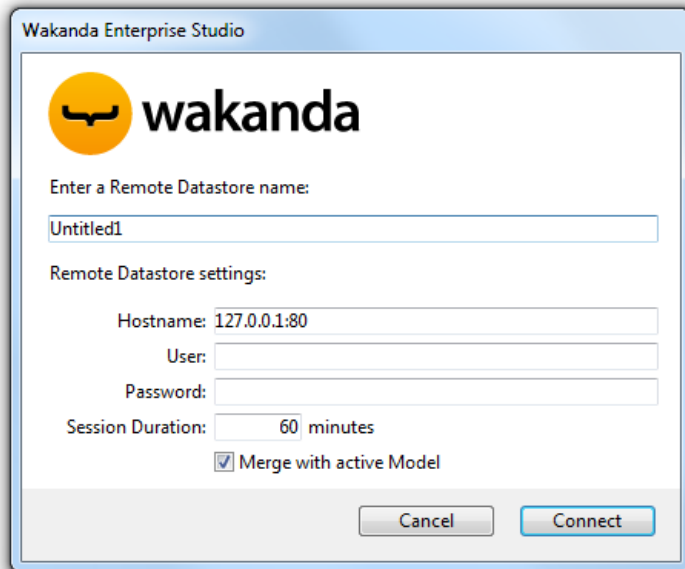


アプリケーションプロジェクトが作成され、Wakanda Studio Explorerのデフォルトの項目がウィンドウの左側に表示されます。

3. PROJECT の行を右クリックし、コンテキストメニュー内から**Connect to Remote Datastore...** コマンドを選択します。



接続ダイアログボックスが表示されます:



4. リンクの名前を入力します。ここでは仮に"Emp4D"と入力します:

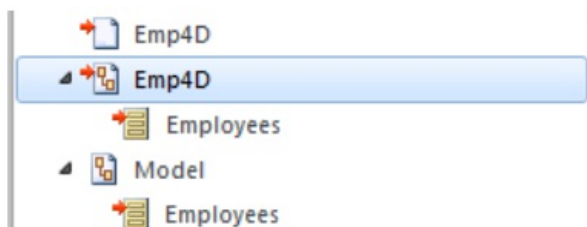
Enter a Remote Datastore name:

Emp4D

"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。

5. (任意) もし4D Server が Wakanda Enterprise Studio と異なるマシンにある場合、Hostname の欄にそのマシンのホスト名もしくはIPアドレスを入力します。同じマシン上にある場合は、ローカルアドレス"127.0.0.1:80"のままにするか、"localhost"と入力します。
6. その他の欄はデフォルトの引数のままで **Connect** ボタンをクリックします。

数秒後、"Emp4D" という外部モデルが Wakanda アプリケーションのファイルの中に表示され、4Dの[Employees] テーブルがローカルモデルのdatastore classの中に表示されます。外部モデルは赤い矢印が表示されます:



注: 接続引数を含んでいるのは一つ目のEmp4Dファイルです。

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ージャーなど)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、4D Mobile サービスが有効化されていて、テーブルが公開されているか。

- "Hostname"へ渡されたアドレスが有効であるか。

4D Serverが実際に4D Mobile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
```

(4D Mobile に公開されている全てのテーブルを返します。)

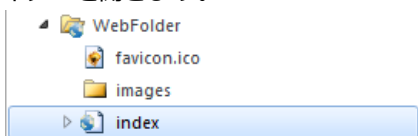
```
<address>/rest/my_table/my_method
```

(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

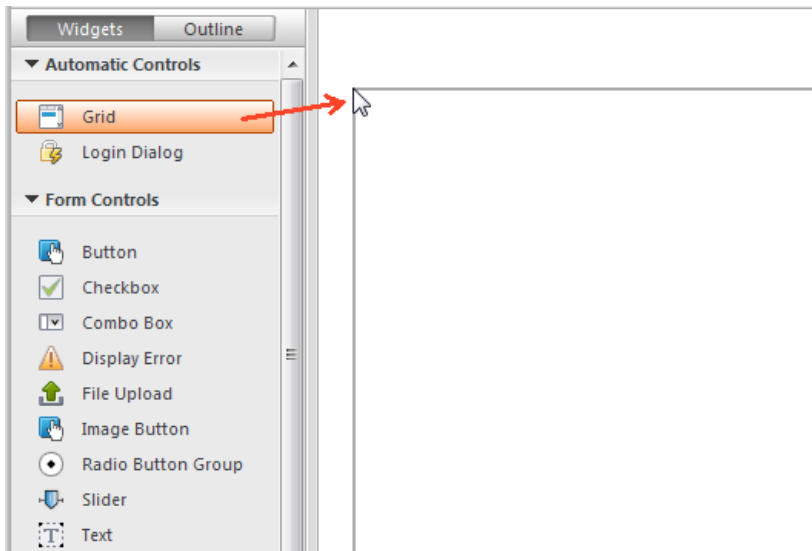
ここでは単純なドラッグ&ドロップによって4DテーブルとWakandaウィジェットを関連づけ、Wakanda Enterprise Serverを起動してデータを表示させます。

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。

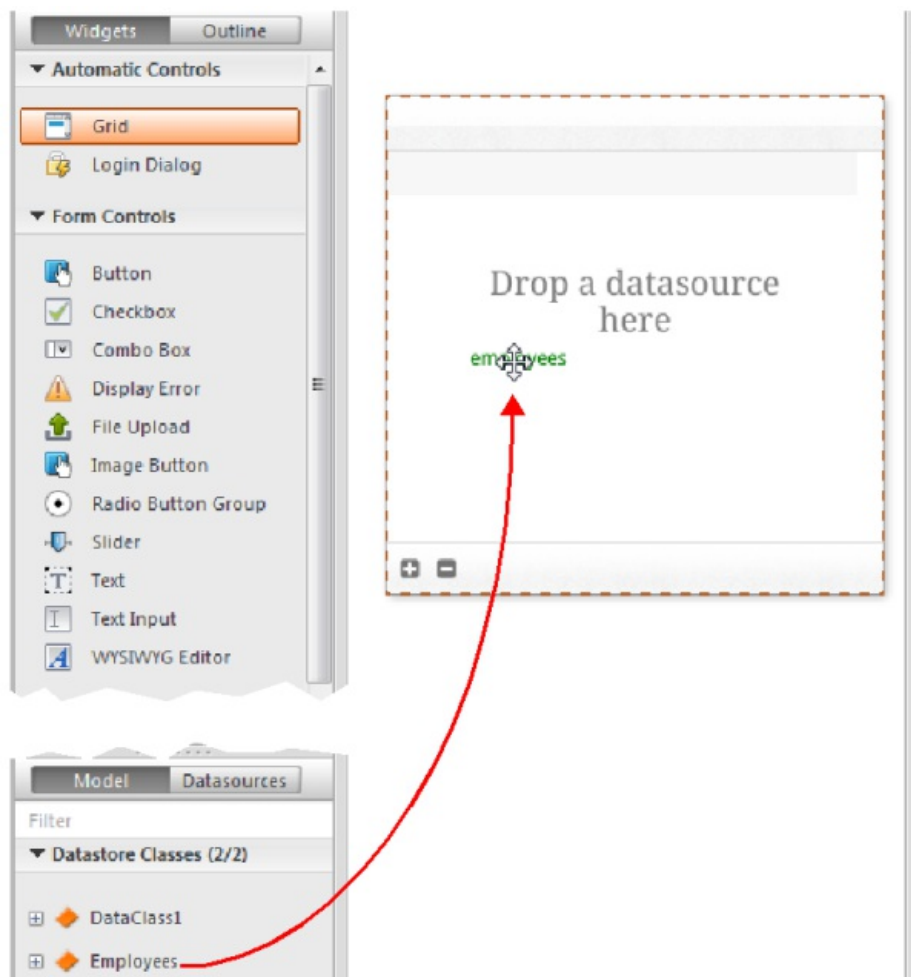


注: "WebFolder"にはプロジェクトの中のWeb 公開に必要な要素が置かれています。"Index"はプロジェクトのデフォルトのページになります。

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:

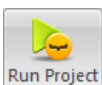


この時点で、エディターは"Employees"クラスをもとにした *datasource* を自動的に作成します。これはウィジェットのコンテンツを管理します。この *datasource* とはWakanda によって管理される JavaScript オブジェクトで、デフォルトでは"employees"という名前がついています(クラス名の頭文字が小文字になったものです)。ウィジェットには中身のプレビューが表示されます。ウィンドウを広げることによってデータソースの全フィールドを表示することができます。

ID	Lastname	Firstname	Salary
Text	Text	Text	Text

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。

- 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。
- 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できます:

ID	Lastname	Firstname	Salary
1	Brown	Michael	25000
2	Jones	Maryanne	35000
3	Smithers	Jack	41000

3 item(s)

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:

ID	Lastname	First name	Salary
1	Brown	Michael	25000
2	Jackson	Maryanne	35000
3	Smithers	Jack	41000

Emp4D - Employees: 3 of 3		
ID :	Lastname :	Firstname :
1	Brown	Michael
2	Jones	Maryanne
3	Smithers	Jack

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。

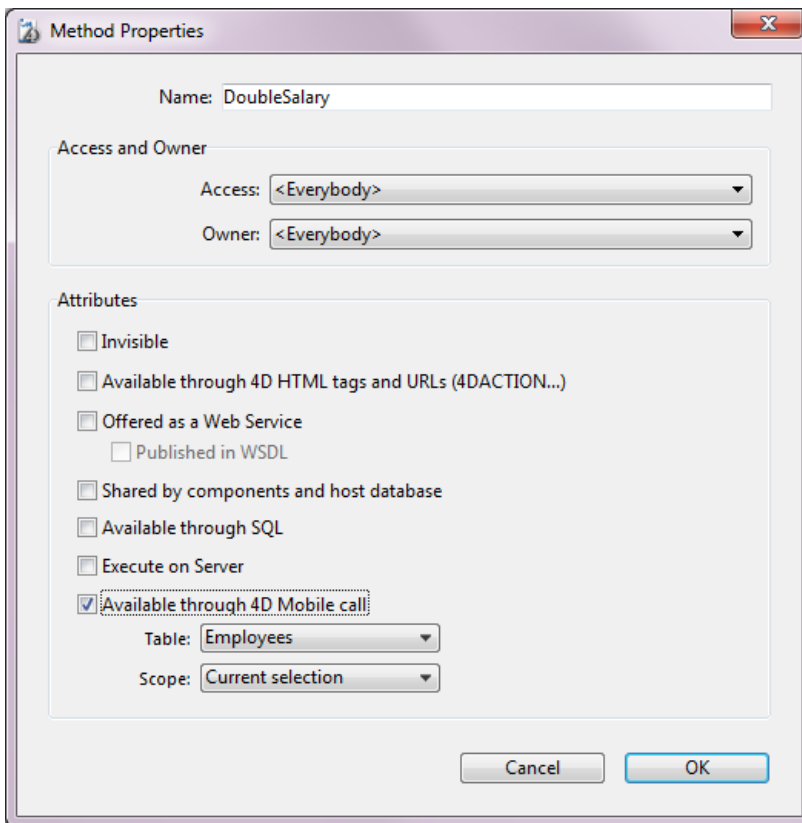
1. 4D 側で、**DoubleSalary** という名前のプロジェクトメソッドを作成し、以下のコードを入力します:

```

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

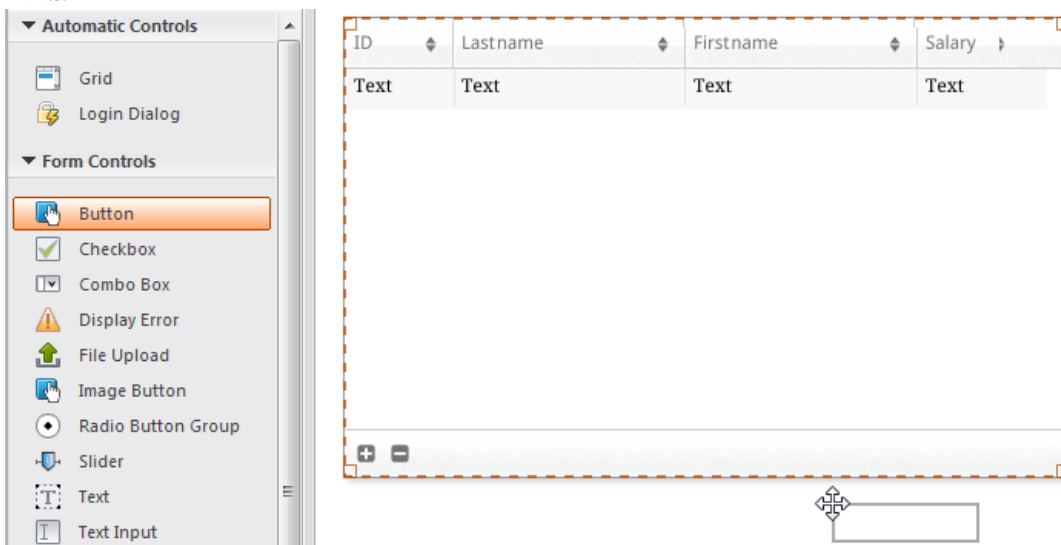
```

2. メソッドプロパティの4D Mobile 呼び出しを以下のように設定し、**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。

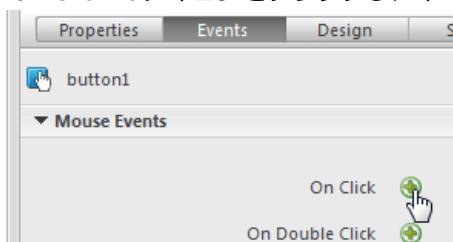
3. Wakanda Enterprise Studio 側で、GUI Designer 中の **Index** に戻り、ウィジェットのリストからボタンを選択して追加します:



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:

Double salaries

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、コールバック関数(*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が:

```
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:

ID	Lastname	Firstname	Salary
1	Brown	Michael	25000
2	Jackson	Maryanne	35000
3	Smithers	Jack	41000

3 item(s)



Salaryの欄の値が倍増したのが確認できます:

ID	Lastname	Firstname	Salary
1	Brown	Michael	50000
2	Jackson	Maryanne	70000
3	Smithers	Jack	82000

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

セキュリティ上・パフォーマンス上の理由から、4D Mobile (Wakanda server) 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:

- 4D Mobile サービスのスタートアップ
- 4D Mobile 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4D Mobile サービス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4D Mobileからアクセス可能
 - プロジェクトメソッドは4D Mobileからアクセス不可

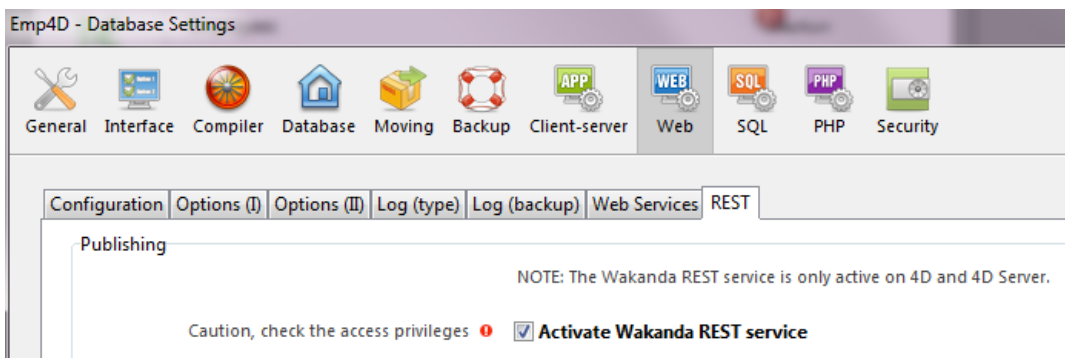
4D Mobileサービスの有効化

デフォルトとして、4D Server は4D Mobile 要求には反応しません。Wakanda/4Dコネクタ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、4D Mobile サービスを有効化する必要があります。

注: 4D Mobile 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。

4D Mobile 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4D Mobile タグをクリックします。
2. **4D Mobile サービスを有効化**のオプションにチェックをします:



Wakanda 4D Mobile 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4D Mobile 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)。

4D Mobile 接続の管理

4D Mobile 接続の管理とは、Wakanda 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。

Wakanda 4D Mobile アクセスの一部としてチェックされる識別子は、以下によって実行された接続リクエストの際に送信された名前とパスワードです:

- Wakanda Enterprise Studioの"Connect to Remote Datastore"ダイアログボックス
- [mergeOutsideCatalog\(\)](#)、[openRemoteStore\(\)](#)、もしくは[addRemoteStore\(\)](#) SSJSメソッド

包括的に4D Mobile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
- **On 4D Mobile Authentication database method**を用いてプログラムによって管理する方法

これらの管理モードはどちらかしか選ぶことができません。On 4D Mobile Authentication database methodが定義されていた場合、4Dパスワードによる自動アクセス管理は無効化されます。

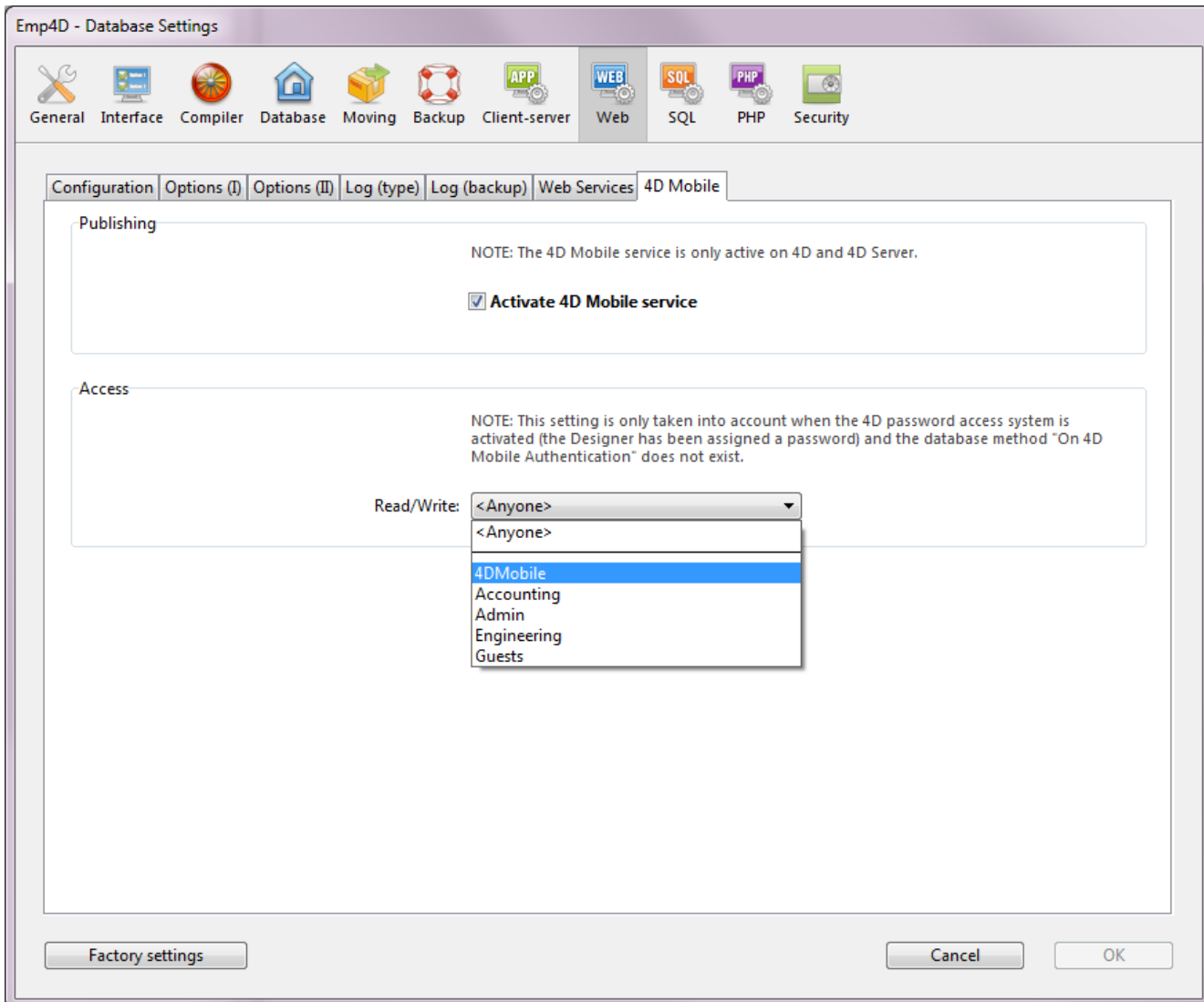
警告：もしこれら二つの管理モードがどちらも有効化されていなかった場合、4D Mobile を通してのデータベースへのアクセスは常に受理されます(これは推奨されません)。

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：

1. データベース設定の画面からWeb→4D Mobile 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4D Mobile 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 リクエストを通して4Dへとアクセスできるようになり、4D Server上で例えば **mergeOutsideCatalog()** メソッド等を使用してセッションを開くことができます。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 4D Mobile Authentication database method** 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 4D Mobile Authentication database methodを使用する方法

On 4D Mobile Authentication database methodを使用することによりWeb Serverエンジンに送られた4D Mobile 接続のアクセス権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4D Mobile 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4D Mobile 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 4D Mobile Authentication database method**の詳細を参照して下さい。

4D Mobileに公開されている 4D オブジェクトの設定

Wakanda 4D Mobile 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4D Mobile 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 Server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、4D Mobile へと公開されません。

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することもできます。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4D Mobile 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サービスで公開」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サービスで公開」にチェックをします。

4D Mobile 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。

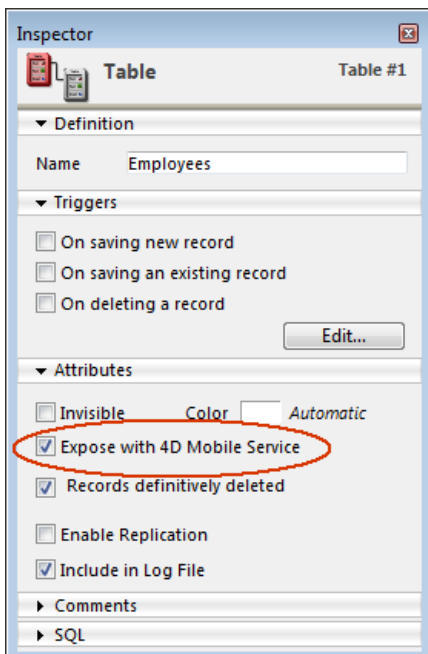
テーブルの公開

デフォルトでは、全てのテーブルは4D Mobile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4D Mobile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が、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4D Mobile 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、**4D Mobile 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. **4D Mobile 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。

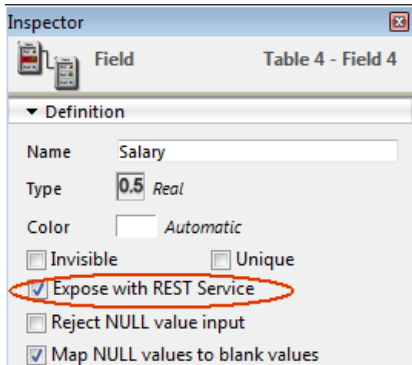
フィールドの公開

デフォルトでは、4Dフィールドは全て4D Mobile 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、4D Mobile 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4D Mobile 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、**4D Mobile 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. **4D Mobile 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4D Mobileで公開するためには、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4D Mobile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**4D Mobile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

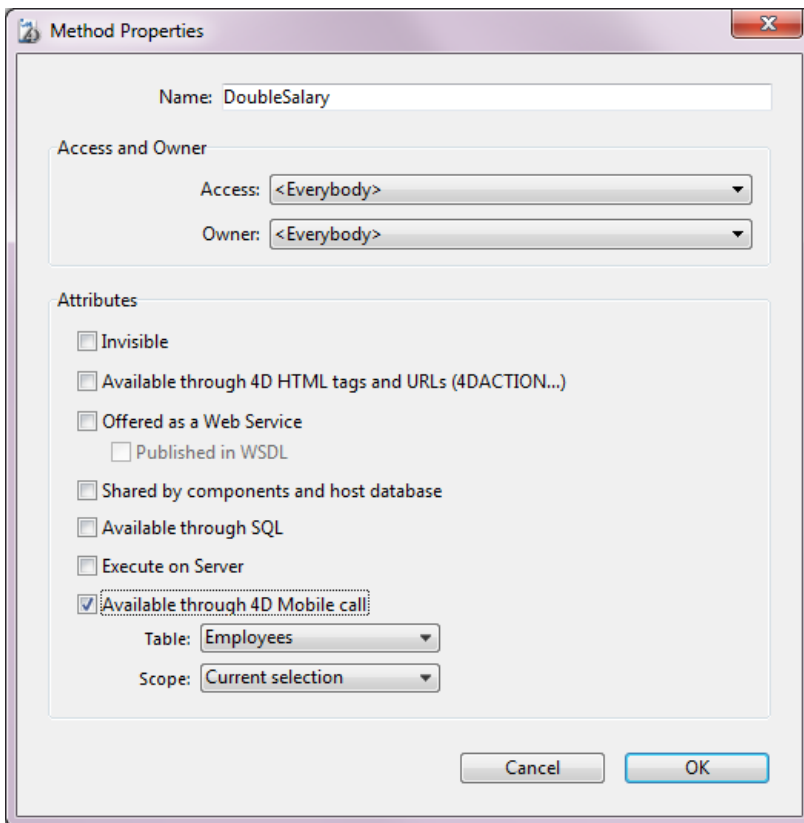
デフォルトではどのプロジェクトメソッドも4D Mobile では公開はされていません。

しかし、場合によっては一部のプロジェクトメソッドを4D Mobile 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Wakanda 実行コンテキストを定義する必要があります。

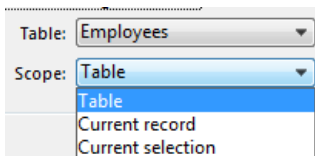
Note: 4D メソッドにアクセスグループが関連付けられている場合、4D Mobile のグループがこのグループに含まれている必要があります。

プロジェクトメソッドの4D Mobile 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. **4D Mobile 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Wakanda 実行コンテキストを定義します。



これらの設定は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。

4D Mobile 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「4D Mobile 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

4D Mobile 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。この設定はテーブルのデータの使用そのものとはリンクしておらず、JavaScriptコードからメソッドへアクセスする際の`datastore class` オブジェクトを指定することができます。
メニューには4D Mobile 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。メソッドがテーブルのデータを特に使用する場合、そのテーブルを選択することができます。メソッドが単一のテーブルに関連付けられていない場合、公開されているどのテーブルも使用することができます。または、ご自分の4Dアプリケーションのビジネスロジックに対応するメソッドのみを公開したい場合、それ専属のテーブルを作成・公開することができます。例えば、[4D MobileInterface] というテーブルを作成し、4D Mobileに公開されている全てのプロジェクトメソッドをそこに関連付ける、といった具合です。
- **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` という型のオブジェクトとして呼び出されます。
例:`ds.MyTable.MyMethod`
 - **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。例:

`ds.MyTable(1).MyMethod`

- **カレントセクション:** 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。

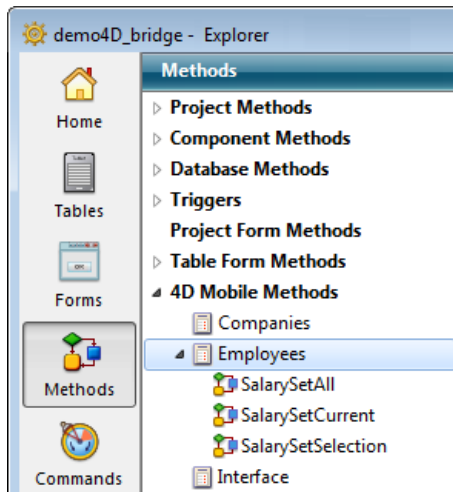
Wakanda 側では、メソッドは *Entity Collection* という型のオブジェクト上で呼び出されます。例:

`ds.MyTable.all().MyMethod`

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

4D Mobile サービスが有効化されているとき、4D Mobile 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の一覧は、4Dエクスプローラー内の4D Mobile メソッドのセクションに表示されます:



🔧 On 4D Mobile Authentication database method

\$1, \$2, \$3 -> On 4D Mobile Authentication database method -> 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 4D Mobile Authentication database method は4D Mobile 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4D Mobile 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 4D Mobile Authentication database method** 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 4D Mobile Authentication database method** が定義される(つまり中にコードが記述される)と、4D は4D Mobile 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/4D Mobile 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 4D Mobile 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。

4D Mobile 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)。

4D Mobile 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 4D Mobile Authentication database method** 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 4D Mobile 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 4D Mobile Authentication database method** 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("4DMOBILE_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、以下の二つの方法を用いて4Dのデータベースへと接続することができます:

- "Connect to Remote Datastore" ダイアログボックスを使用する(Wakanda Enterprise Studio内にあります)。
- JavaScript メソッド(`mergeOutsideCatalog()`, `openRemoteStore()` or `addRemoteStore()`)を実行する。

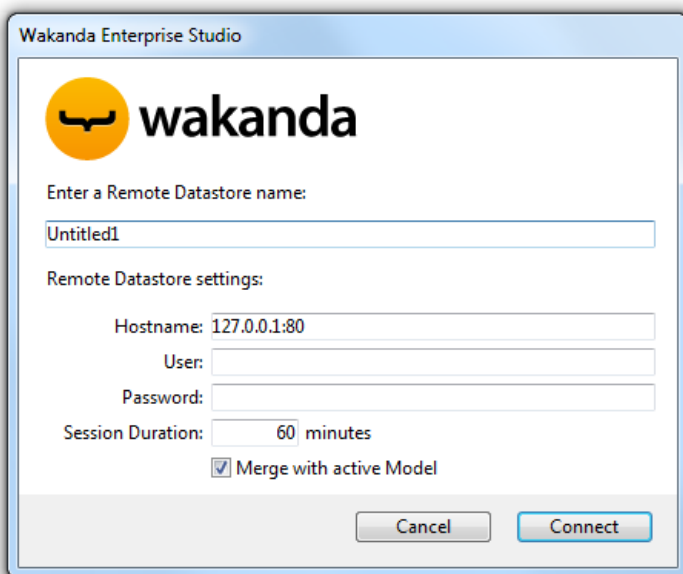
Wakandaと4Dの間に接続が確立されると、Wakandaは4Dアプリケーションにて公開されているテーブル、属性、そしてプロジェクトメソッドをローカルオブジェクトとして使用する事ができるようになります。

また他のJavaScriptコードを実行する事もできます。例えば、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、クラスを拡張したり、計算属性を追加したりできます。

Connect to Remote Datastoreダイアログボックスを使用して接続

Wakanda Enterprise Studioでは、**Connect to Remote Datastore...** コマンド(**File** メニューまたはプロジェクトのコンテキストメニュー内にあります)を使用してリモートのdatastore とのリンクを開設します。このリモートのdatastore としては4Dデータベースまたは他のWakandaアプリケーションを使用可能です。どちらの場合でも、Wakanda Enterprise Studio がリモートモデルにアクセスできるようにするために、リモート datastore のHTTPサーバーは開始される必要があります。リンクが定義されると、アプリケーションが開かれるたびに".waRemoteConfig"ファイルに保存された接続引数を使用して自動的に復元されます(以下を参照して下さい)。

Connect to Remote Datastore... コマンドを選択すると、接続ダイアログボックスが表示されます:



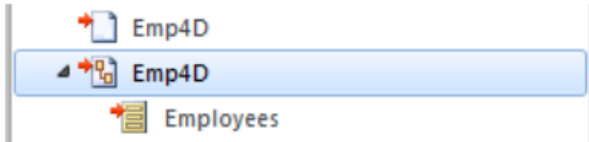
この中では以下の接続引数を設定する事ができます。

- **Remote datastore name:** Solution Explorerに表示されているリモートカタログのローカル名を指定します。**Merge with active Model** オプションのチェックを外すと、この名前は"ds" としてではなく、datastore id として使用されます(以下を参照して下さい)。この場合、使用可能な文字のみを使用するように気を付けて下さい(Wakanda ドキュメント内の[Programming and Writing Conventions](#) を参照して下さい)。
- **Hostname:** リモートデータサーバーのアドレスを指定します(より良い安全性のためにHTTPSを使用して下さい)。
- **User** と**Password:** 4Dデータベース上で4D Mobileセッションを開くためのユーザー名とパスワードを指定します。
- **Session duration:** リモート4Dデータベースへと接続したセッションを保持する分数を設定します(デフォルトは60)。この引数は接続がユーザー名と空でないパスワードによって開かれていた場合にのみ有効になります(4D Mobile 接続を4D側で保護する事は強く推奨されています)。
- **Merge with active Model** (デフォルトではこのオプションはチェックされています): このオプションをチェックす

ると、リモート *datastore classes* が **ds** ネームスペース(特にWakanda のGUI Designerのクラスのリスト)に表示されるようにするために、リモート *datastore with* をプロジェクトのアクティブなモデル(**ds** オブジェクト)と統合します。より詳細な情報に関しては、[アクティブなモデルとの併合か、専属のモデルの使用か](#)を参照して下さい。

Parameter ファイル

"Connect to a Remote Datastore"ダイアログボックスを使用してWakanda と 4D Server との間の接続が確立されると、Wakanda Enterprise Studio は自動的にプロジェクトのフォルダ内に二つのファイルを作成します(作成されたファイルはアイコンに赤の矢印が付いています)。



- 最初のファイル(拡張子が".waRemoteConfig"のファイル)には、ダイアログボックス内で定義された接続パラメーターが保存されます。
- 二つ目のファイル(拡張子が".waRemoteModel"のファイル)には、リモートのdatastoreのモデルをローカルに表したものが含まれます。この中身はWakanda モデルエディターウィンドウ内で表示する事ができます(ただし編集はできません)。

Note: Youファイルの拡張子はWakanda Studio の Explorer 内でファイルを選択したときに表示されるヘルプTipの中に表示されます。

JavaScript メソッドを使用して接続

Wakanda Enterprise Server では、JavaScriptメソッドを実行することによっても 4D データベースとのリンクを設定する事ができます。各セッション中にリンクを有効にするためには、接続メソッドは通常アプリケーションが開かれたときに実行されるコード(bootstrap.js)の中か、またはモデルが開かれたときに実行されるコード(model.js)内に置かれる必要があります。

4D Mobile リンクを確立するために使えるメソッドは三つあります:

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

これらのメソッドの主な違いは、リモートの *datastore* から来るオブジェクトとWakandaアプリケーションとの統合のされ方に関係します。**`model.mergeOutsideCatalog()`** は、リモートのカタログをアクティブなモデルと併合させます。その一方で、**`addRemoteStore()`** と **`openRemoteStore()`** は専属のモデルを生成します。この点についての詳細は、以下の[アクティブなモデルとの併合か、専属のモデルの使用か](#)を参照して下さい。

`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 モデルと併合させます。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
	文	
localName	文字列	リモートカタログのローカル名
	文	
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)
	文	
userName	文字列	セッションを開くためのユーザー名
	文	
password	文字列	セッションを開くためのパスワード
	文	
jsFilePath	文字列	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名(任意、詳細は 外部ファイルの変更を参照して下さい)
	数字	
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)この引数は接続がユーザー名と空でないパスワードで開かれている場合にのみ有効であるという点に注意して下さい(4D Mobile接続を4D Server側で保護する事は強く推奨されています)。

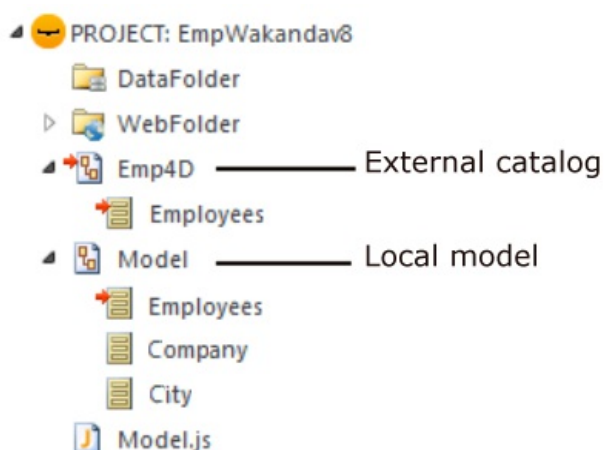
より詳細な情報に関しては、[Wakanda Server-Side API manual](#) の、[mergeOutsideCatalog\(\)](#) メソッドのドキュメントを参照して下さい。

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既に含まれる場合、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**mergeOutsideCatalog()** メソッドを使用したときにローカルのモデルと組み込みされます。

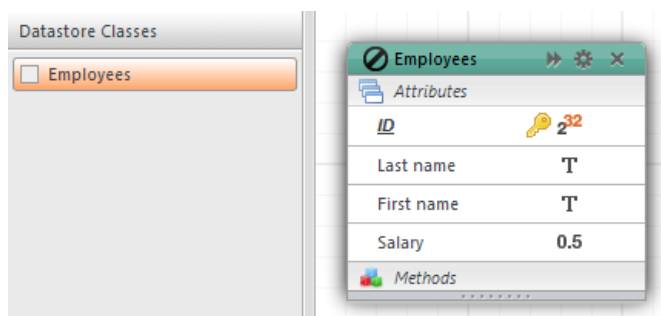
接続が正常に確立されると、「公開」されている4D テーブルがWakanda側のモデルのクラスのリストに表示されます。Wakanda Enterprise Studio 側では、リモートテーブルがローカルモデルのクラスの一覧の中に表示されます。外部要素は赤い矢印で表示されます。

外部カタログはWakanda Studio側でも特定のカタログ(*localName.waRemoteCatalog* という名前)で表示されます。これもまた赤い矢印が表示されています:



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:



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。

アクティブなモデルとの併合か、専属のモデルの使用か

どちらの方法(Wakanda Studio の"Connect to Remote Datastore" ダイアログボックス、または JavaScript メソッドの実行)を使用して4D datastore に接続したとしても、リモートクラス(テーブル)がアクティブなモデルと併合されるか、専属のモデル内に配置されるかのどちらかを選択しなければなりません。

具体的な手法をまとめると以下の表のようになります:

4D datastoreへ接続する手段	アクティブなモデルと併合するには	専属のモデルを使用するには
"Connect to Remote Datastore" ダイアログボックス	Merge with active Model にチェックを入れる	Merge with active Model のチェックを外す
JavaScript メソッド	mergeOutsideCatalog()を使用	openRemoteStore() または addRemoteStore()を使用

アクティブなモデルとの併合

リモート4Dテーブルをアクティブなモデルと併合するとき、アプリケーションの(datastore が ds オブジェクトである)デフォルトのモデルへと、ローカルクラスとして統合されます。データにアクセスする際は以下の原理に沿います。

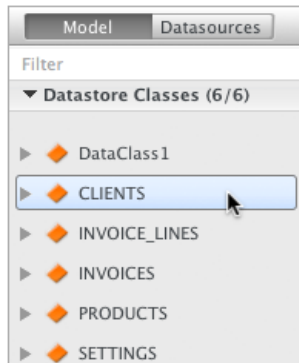
- サーバー側からは、ds オブジェクト(4Dテーブルとメソッドの呼び出しを参照のこと)を使用してリモート4Dテーブルとメソッドへとアクセスします:

```
var invoiceList = ds.INVOICES.all(); //カタログのINVOICESテーブルへとデフォルトでアクセスする
```

- クライアント側からは、Wakanda Ajax Framework (WAF) の自動機能を使用します。リモート4Dテーブルへは、高レベル datasource オブジェクト、または低レベルアクセスを管理する dataprovider API を使用することによってアクセ

スします。

- Wakanda Enterprise Studio では、4D データベースのテーブルはWakandaのGUIデザイナーの中に、ローカルクラスとともに表示されています:



これらの原理は4D Mobileアプリケーションの開発を容易にしてくれますが、それと同時にテーブル間での名前の問題を引き起こすことがあります。これは特にWebアプリケーションが複数の datastore を呼び出した場合に起こります。この場合、リモートの要素を専属のモデル内に配置するのが有効な場合があります。

専属のモデルを使用

リモートの4Dテーブルは、アクティブなモデルと併合しないときには"専属"のモデルを使用します。リモートクラスは、アプリケーションが接続しているdatasotre特有のネームスペースを使用し、これらはdsオブジェクト内部ではアクセスすることができません。この方法なら同じ名前を持つ複数のテーブルを、複数の異なるdatastoreにて使用することが可能になります。

- サーバー側では、**Remote datastore name** 接続引数(ダイアログボックス)または *localName*(JavaScript メソッド)に渡した名前のカスタムのカatalogを使用してリモート4Dテーブルとメソッドにアクセスします。例えば、"my4Dstore"という名前のリンクを作成した場合、アプリケーションのコード内には以下のように記述します:

```
var invoiceList = my4Dstore.INVOICES.all(); // my4Dstore datastore の INVOICESテーブルへアクセス
```

しかしながら、この原則にはカレントのバージョンのWakanda Enterpriseにおいてはいくつかの制約もあります:

- クライアントアプリケーションから、WAFライブラリーやRESTを使用してリモートクラスへと直接アクセスすることはできません。
- リモートクラスは、Wakanda Enterprise Studio の *GUI Designer* の一覧には表示されません。

ですから、クライアントアプリが4Dリモートテーブルのデータへと直接アクセスしないといけないときには、通常 *datastore* には併合モードを使用することが推奨されます。

外部ファイルの変更

Wakanda Enterprise を使用すると、カスタマイズやセキュリティ、最適化などの目的で、外部モデルのローカルバージョンのいくつかの特性を変更することができます。

そのためには、ローカル名と同じ名前と ".js" 拡張子を持つ.jsファイル内に適切なJavaScriptコードを書き、そのファイルをモデルと同じフォルダの中に置きます。例えば、ローカルカタログの名前が *Emp4D.waRemoteModel* であれば、モデルと同じフォルダ内にある *Emp4D.js* という名前のファイルを使用する必要があります。

注:

- v11以降、このファイルはWakanda Studioによって自動的に作成されます。
- JavaScript メソッドを使用して接続を確立した場合、*jsFile* 引数を使用して他の名前を使用することも可能です。

Wakanda は、外部カタログが初期化されたときにこのファイルを使用します。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.calcAtt = new Attribute("calculated", "string"); model.className.calcAtt.onGet = function(); model.className.calcAtt.onSet = function();
```

- エイリアス属性をデータストアクラスに追加することができます。以下の様に記述します:

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

- 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")
```

- 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。

許可の定義

リモートモデルに対して、Wakanda Server 全体に特定の許可を設定することができます。また、それぞれのクラスに関しては個別に許可を設定することができます。この点に関する詳細な情報についてはWakandaドキュメントの [Assigning Group Permissions](#) を参照して下さい。

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から参照されている4Dテーブルへのアクセスモードは、外部カタログがどのように統合されているかによって決まります。これはWakandaがリモートアプリケーションへと接続する際にWakanda内にて定義されます(**アクティブなモデルとの併合か、専属のモデルの使用か**を参照して下さい):

- アクティブなモデルと統合(デフォルトのオプション): この場合、リモートテーブルは、**ds** オブジェクトを通してlocal classと全く同じ様に使用されます。
- 専属のモデルを使用: この場合には、リモートテーブルは専属のモデルオブジェクトのプロパティとなります。

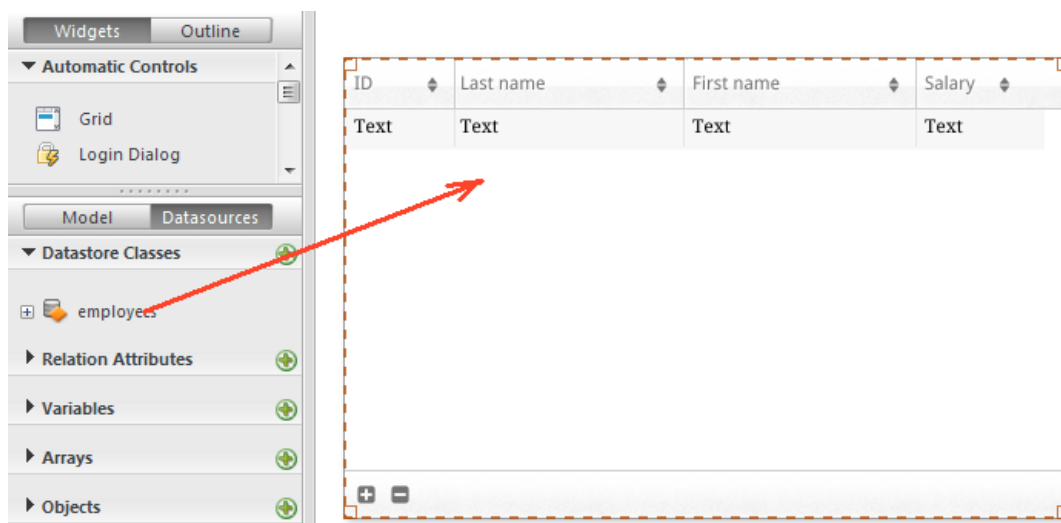
アクティブなモデルと統合されたテーブルの場合

アクティブなモデルと統合した場合、Wakandaアプリケーションから参照されている 4D テーブルは、ローカルの*datastore classes* と同様に、サーバー側のJavaScriptのコードの中で**ds** オブジェクトのプロパティとして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*datastore*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るとemployeesのリストが自動的に表示されます:



ID	Lastname	Firstname	Salary
1	Brown	Michael	25000
2	Jones	Maryanne	35000
3	Smithers	Jack	41000

3 item(s)

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。
```

datastore クラスの仕様については、[Wakanda documentation](#)を参照して下さい。

専属のモデルに置かれたテーブルの場合

参照されている4Dテーブルは、リンクが作成された時点でのカタログのプロパティとして、サーバー側のJavaScriptコード内にて使用されます。このカタログの名前はWakanda Studio 接続ダイアログボックスでの接続引数として**Remote datastore name** に渡したものの、またはJavaScriptメソッドでの *localName* に渡したものになります。

例えば、"my4Dstore" という名前のリンクを作成して、[Employees] テーブルのレコード内にてクエリを実行したい場合、以下の様に記述します:

```
var emp2 = my4Dstore.Employees.query("age > :1", 30); // "my4Dstore"という名前のリンク内で // Employees テーブルのレコード内を検索
```

実装に関する注意: 現在の4D Mobileでは、専属のモデルを使用した場合、クライアント側ではリモートクラスへとアクセスすることはできません。

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) // raiseSalary はdatastore class のプロパティ //カタログはアクティブなモデルと  
統合 var vTot2 = my4DStore.Company.first().capital(param) // first() はentityを返すので、capital はentity  
propertyとなる // my4DStore という専属のモデルを使用
```

- 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:
実装に関する注意: *In the*カレントのバージョンの *Wakanda Enterprise* では、クライアントからの4Dデータベースメソッドへのアクセスは、リモートデータベースがアクティブなモデルと接続・統合されている場合にのみ可能です。
 - [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4D Mobile経由で公開し、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);
```

MOBILE Return selection コマンドを使って、4D セレクションを直接 Wakanda のコレクションとして返すことも可能です。例えば:

```
//FindCountries プロジェクトメソッド
//FindCountries( string ) -> object

C_TEXT($1)
C_OBJECT($0)
QUERY([Countries];[Countries]ShortName=$1+"@")
$0:=MOBILE Return selection([Countries])
```

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。
注:最適化のため、また不要なロックを避けるため、レコードは読み込みのみモードでロードされます。しかしながら、テーブルは読み書き可能なモードなので、**LOAD RECORD** コマンドを呼び出せば、レコードを強制的に読み書きモードでロードすることができます。
- 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4D Mobile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())'`.

MOBILE Return selection (aTable) -> 戻り値

引数	型	説明
aTable	テーブル	カレントセクションを取得したいテーブル
戻り値	Object	Wakanda準拠のセクション

説明

MOBILE Return selection コマンドは、*object* 内に、*aTable* のカレントセクションをWakandaに準拠したentity collectionへと変換したものを、JSONオブジェクトとして返します。

このコマンドは、4D Mobile接続(通常はRESTを経由した4DとWakanda間の接続)のコンテキストにおいて呼び出されることを想定しています。4D Mobile接続が確立され適切なアクセス権が設定されると、Wakandaは\$0 引数に値を返す4Dプロジェクトメソッドを実行する事ができます。

MOBILE Return selection コマンドは、*aTable* で指定したテーブルのレコードのカレントセクションを、JSONフォーマットの *entity collection* オブジェクト形式で\$0 引数に返します。このオブジェクトはWakandaでレコード(または*entities*)のセクションを内包するentity collectionsに準拠しています。

4D Mobileアクセスのためには、4Dデータベース内において、以下の特定の設定をしなければならないことに注意して下さい:

- Webサーバーが起動している必要があります。
- データベース設定内にて、"4D Mobile サービスを有効化"のオプションがチェックされているかどうかを確認して下さい。
- 有効なライセンスが必要になります。
- 公開したテーブルとフィールドがどちらも"4D Mobileサービスで公開"のオプションにチェックがされていなければなりません(デフォルトではチェックがされている)。
- 呼び出されるメソッドは、"4D Mobile からの利用を許可"のオプションにチェックがされている必要があります(デフォルトではチェックされていません)。

aTable には、有効なテーブルであればデータベース内のどんなテーブルでも渡す事ができ、メソッドプロパティにてテーブルと関連付けがなされているテーブルに限らないという点に注意して下さい。この引数はメソッドが呼び出し可能なオブジェクトをWakanda側で判断するためにのみ使用されます。

4D Mobileの設定についての詳細な情報に関しては、[4D Mobile](#)ドキュメントを参照して下さい。

例題

[Countries] テーブルのクエリに基づいたカレントセクションを Wakanda のグリッドに表示させたい場合を考えます。

まず、以下の様な4Dメソッドを作成します:

```
//FindCountries プロジェクトメソッド
//FindCountries( 文字列 ) -> object

C_TEXT($1)
C_OBJECT($0)
QUERY([Countries]; [Countries] ShortName=$1+"@")
$0:=MOBILE Return selection([Countries])
```

返されたセクションは有効なコレクションとして、Wakanda 内で直接使用する事ができます。

4D Mobileを経由して4Dと接続しているWakanda Serverにおいて、4DのCountries tableと関連付けられたグリッドを持つページを作成したとします。デフォルトでは、ランタイムでは、4D テーブルからの全てエンティティが表示されています:

ShortName	Name	Capital
Angola	Republic of Angola	Luanda
Argentina	Argentine Republic	Buenos
Australia	Commonwealth of Au...	Canberr
Brazil	Federative Republic of...	Brasilia
Canada	Canada	Ottawa
Chile	Republic of Chile	Santiago
China	People's Republic of C	Beijing

24 item(s)

Find Countries

ボタンに記述されているコードは以下の通りです:

```
button1.click = function button1_click (event) {
    sources.countries.FindCountries("i", { //4Dメソッドを呼び出し。"i" は$1として渡されます。
        onSuccess:function(coll){ //コールバックファンクション(非同期)。$0を引数として受け取ります。
            sources.countries.setEntityCollection(coll.result); //カレントのエンティティコレクションを// coll.resultオブジェクト内のものと置き換えます
        }
    });
};
```

その結果、グリッドが更新され以下の様になります:

ShortName	Name	Capital
India	Republic of india	New D
Italy	Italian Republic	Rome

2 item(s)

Find Countries

Selected record number

Selected record number {(aTable)} -> 戻り値

引数	型	説明
aTable	テーブル	→ レコード位置番号を取得するテーブル、または 省略時はデフォルトテーブル
戻り値	倍長整数	↩ カレントレコードのレコード位置番号

説明

Selected record numberは、*aTable*のカレントセレクション内でのカレントレコードの位置を返します。

セレクションが空ではなく、カレントレコードがそのセレクションに含まれるときに、**Selected record number**は1から**Records in selection**までの値を返します。セレクションが空かカレントレコードが存在しない場合、この関数は0を返します。

レコード位置番号は、**Record number**で求めるレコード番号とは異なります。**Record number**は絶対レコード番号を返します。レコード位置番号は、カレントセレクションおよびカレントレコードに依存します。

例題

以下の例は、カレントレコードのレコード位置番号を変数に格納します：

```
CurSelRecNum:=Selected record number ([People]) `レコード位置番号を取得
```

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:

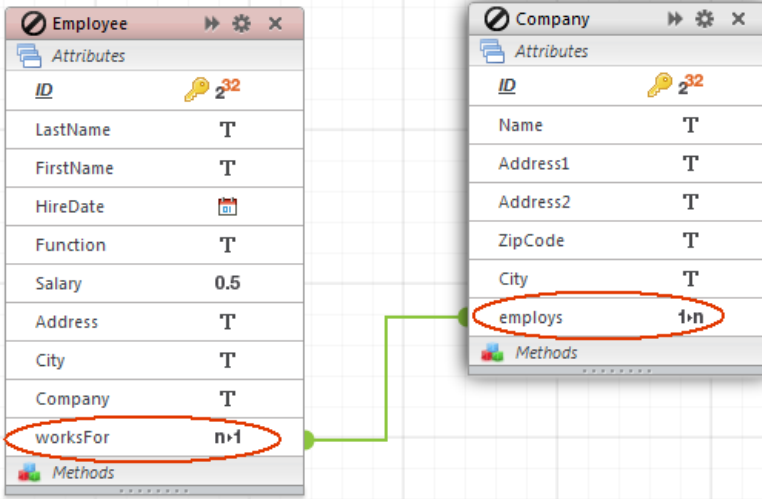
- リレーションのソーステーブル(クラス)内にて、n->1属性
- リレーションのデスティネーションテーブル(クラス)にて、1->n属性

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できます。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できます:

The screenshot displays the Wakanda Model Editor interface. On the left, two tables are shown: 'Employee' and 'Company'. The 'Employee' table has attributes: ID (primary key), LastName, FirstName, HireDate, Function, Salary, Address, City, and Company. The 'Company' table has attributes: ID (primary key), Name, Address1, Address2, ZipCode, and City. A relationship is established between the 'Company' attribute of the 'Employee' table and the 'Name' attribute of the 'Company' table. On the right, the 'Inspector' window shows the configuration for this relationship. The 'From' field is '[Employee]Company' and the 'To' field is '[Company]Name'. The 'Color' is set to 'Automatic'. Under the 'Many to One Options' section, the 'Name' is 'worksFor'. Under the 'One to Many Options' section, the 'Name' is 'employs'. Both 'worksFor' and 'employs' are circled in red in the original image.

Wakanda側では、コネクタ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できます:



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:

The screenshot shows the Wakanda IDE interface. On the left, the 'Datstore Classes (2/2)' panel shows the 'Company' class with its attributes and the 'employs' relationship. The 'Employees' class is also visible. In the center, two data grids are shown: 'Companies' with columns ID, Name, and Address1; and 'Employees' with columns ID, LastName, and FirstName. Red arrows point from the 'Company' class in the left panel to the 'Companies' grid, and from the 'employs' relationship to the 'Employees' grid. The bottom status bar shows 'waf-body' and 'dataGrid1'.

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セッションの管理

概要

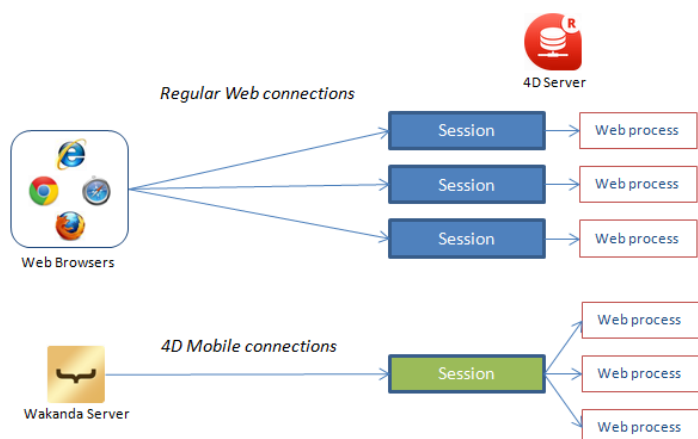
4D v15 R4以降、プログラミングを使用して4D MobileセッションのIDに4D Server側からアクセスできるようになりました。この機能によって、開発者はセッションに関連した情報をローカルに取得あるいは設定できるようになりました(以下の例題を参照して下さい)。

4D Mobileセッションは通常の4D Webセッションコマンドを使用して管理されます。複数の4D Webコマンドに加え、**WEB Get session process count**コマンドと[#title id="2779"/])も4D Mobileセッションをサポートします。

4D Mobile セッション vs Web セッション

4D MobileセッションとWebセッションは二つの異なる種類のセッションです。両者では一部の概念(とコマンド)が共通していますが、プロパティが異なります。主な違いとしては、セッション、プロセス、そしてプロセスコンテキスト間の関係性が挙げられます:

- Webセッションは単一のWebプロセスとリンクされます(一対一の関係にあります)。自動セッション管理機能により、プロセスコンテキスト(変数のインスタンス、セクション、等)は再度利用する事が可能です。
- 4D Mobileセッションは複数のWebプロセスとリンクする事が可能です。それぞれのプロセスコンテキストは、プロセスメソッドの実行終了時に自動的にリセットされます。



結果として、4D Mobile Webプロセス間のセッションに関連した情報を共有するためには、4D Server側での特定の実装をする必要があります。

何の変更も加える事なくサポートされるコマンド

以下の既存のWebセッション管理コマンドは4D Mobileセッションをサポートします。

WEB CLOSE SESSION(sessionID)

WEB CLOSE SESSIONコマンドはsessionID引数にIDを渡された4D Mobileセッションを閉じます。4D Mobileセッションは複数のプロセスを管理できるため、このコマンドは実際には関連した全てのWebプロセスの実行の終了をリクエストします。

WEB Get Current Session ID -> sessionID

WEB Get Current Session IDコマンドは、カレントの4D Mobileセッションに関連付けられたUUIDを返します。

WEB GET SESSION EXPIRATION (sessionID ; expDate ; expTime)

WEB GET SESSION EXPIRATIONコマンドは4D Mobileセッションのcookieに関連した失効情報を返します。

同じcookieは、4D Mobileセッションに関連した全てのプロセスに対して使用されます。

新しいWEB Get session process countコマンド

WEB Get session process countコマンドは、特定のセッションに関連した既存のプロセスの数を調べます。

- 通常のWebセッションに対しては、コマンドは常に1を返します(1Webセッション=1プロセス)
- 4D Mobileセッションに対しては、コマンドは関連したWebプロセスを全て返します。このコマンドは、例えばこのコンテキストにおいて4D Mobileセッションの全てのプロセスに対してループを実行したい場合などに有効です。

On Web Close Process データベースメソッド(旧 On Web Session Suspend)

On Web Close Process データベースメソッドは、Webプロセスがその実行を終えようとするたびに4Dによって呼び出されます。これは4D Mobileセッションプロセスを完全にサポートします。このコンテキストにおいて、閉じられたWebプロセス毎に呼び出されるので、4D Mobileセッションプロセス中に生成されたどのようなデータ(変数、セレクション等)も保存する事ができます。

注: 通常のWebセッションについては、Webセッション(つまりWebセッションユニークプロセス)が閉じられるたびに、**On Web Close Process データベースメソッド**データベースメソッドが呼び出されます。

例題

単一の4D Mobileセッションの複数のプロセス間での情報を共有あるいは再利用したい場合、4D MobileセッションのUUIDを使用してセッション関連情報を指定する事ができます。例えば、レコードをクエリしたあと、命名セレクションを4D Serverに保持しておくことで、同じセッション内でのその後のRESTリクエストがそのセレクションに直接アクセスできるようにしたい場合を考えます。クエリ宣言後に、以下のように書く事ができます:

```
//セッションUUIDを含んだインタープロセスセレクションを作成
COPY NAMED SELECTION ([Emp]; "<>EmpSel"+WEB Get Current Session ID)

//あとでそのセッションからのセレクションを再利用することが可能
USE NAMED SELECTION ([Emp]; "<>EmpSel"+WEB Get Current Session ID)
```

WEB Get session process count

WEB Get session process count (sessionID) -> 戻り値

引数	型		説明
sessionID	テキスト	→	セッションUUID
戻り値	倍長整数	↩	セッションに関連づけられたプロセスの数

説明

WEB Get session process count コマンドは、*sessionID* 引数に渡したUUIDを持つセッションに関連づけられた実行中のプロセス数を返します。

このコマンドは、4D v15 R4で導入された**4D Mobileセッションをプログラミングで管理**機能のコンテキストで追加されたものです。主に4D Mobileセッションによって実行されているプロセスの数を数えるために設計されています。

- 4D Mobile セッションに対しては、このコマンドは実際のプロセス数を返します。4D Mobileセッションは複数のプロセスが実行可能です。
- 通常のWebセッションに対しては、このコマンドは常に1を返します(1Webセッションに対し1プロセスが一体化します)。

例題

情報をカレントの4D Mobileセッション上で配列形式で保存したい場合を考えます:

```
C_TEXT($sessionID)
C_LONGINT($count)
C_DATE($expDate)
C_TIME($expTime)

$sessionID:=WEB Get Current Session ID
$count:=WEB Get session process count($sessionID)
WEB GET SESSION EXPIRATION($sessionID;$expDate;$expTime)

APPEND TO ARRAY($aTimestamp;String(Current date)+" "+String(Current time))
APPEND TO ARRAY($aSessionUID;$sessionID)
APPEND TO ARRAY($aNbProcesses;$count)
APPEND TO ARRAY($aExpirationDate;$expDate)
APPEND TO ARRAY($aExpirationTime;$expTime)
```

WEB CLOSE SESSION (sessionID)

引数	型	説明
sessionID	テキスト	セッションUUID

説明

WEB CLOSE SESSION コマンドは *sessionID* 引数で指定された既存のセッションを破棄します。指定されたセッションが存在しない場合、コマンドはなにもしません。

Webプロセスや他のプロセスからこのコマンドが呼び出されると:

- ブラウザーに送信されるcookieの有効期限が0に設定されます。
- 開発者がセッション情報を保存できるようにするために **On Web Close Process データベースメソッド** が呼び出されます。
- カレントセレクションや変数などのプロセスオブジェクトが消去され、レコードのロックが解除されます。

このコマンド実行後、Webクライアントが当該cookieを使用して4D Webサーバーにアクセスすると、新しいセッションが開始され新しいcookieがクライアントに送信されます。

注: 4D Mobileセッションのコンテキストにおいては、**WEB CLOSE SESSION** コマンドは *sessionID* 引数に渡したIDの4D Mobileセッションを閉じます。4D Mobileセッションは複数のプロセスを管理できるため、このコマンドはセッションに関連した全てのWebプロセスに対して実行を終了を要求します。

WEB Get Current Session ID

WEB Get Current Session ID -> 戻り値

引数	型		説明
戻り値	テキスト		セッションUUID

説明

WEB Get Current Session IDコマンドはカレントのWebリクエストのセッションIDを返します。このIDは4Dが自動で生成します。

このコマンドがWebセッション管理のコンテキストの外で呼び出されると、コマンドは空の文字列を返します。

WEB GET SESSION EXPIRATION

WEB GET SESSION EXPIRATION (sessionID ; expDate ; expTime)

引数	型		説明
sessionID	テキスト	→	セッションUUID
expDate	日付	←	cookie有効期限日
expTime	時間	←	cookie有効期限時刻

説明

WEB GET SESSION EXPIRATION コマンドは *sessionID* に渡された UUID のセッションの cookie の有効期限に関する情報を返します。

expDate 引数は cookie の有効期限日を、 *expTime* 引数は cookie の有効期限時刻を受け取ります。

注: Webレスポンスがクライアントに送信されるたびに、cookieの有効期限はリクエストが行われた時刻+Web Inactive session timeout (デフォルトで8時間) に設定されます。例えばデフォルト値の状態で:

最初のリクエスト: 月曜日の1:00

-> 有効期限は月曜日の09:00

二番目のリクエスト: 月曜日の1:10

-> 有効期限は月曜日の09:10

三番目のリクエスト: 火曜日の4:00 (cookieの有効期限が過ぎている)

-> 新しいcookie値が生成され、有効期限は火曜日の12:00

🔧 On Web Close Process データベースメソッド

On Web Close Process データベースメソッド

このコマンドは引数を必要としません

On Web Close Process データベースメソッド はWebセッションが閉じられる直前に、4D Webサーバーから呼び出されます。4Dは以下のような場合にWebセッション (セッションを管理するWebプロセス) を閉じます:

- セッションを管理するWebプロセス数の最大値 (デフォルトで100、**WEB SET OPTION**コマンドで変更可能) に達している状態で、さらに新しいWebセッションを作成する必要があるとき (4Dが一番古いWebセッションプロセスを自動で破棄します)
- セッションプロセスのタイムアウトに達したとき (デフォルトで480分 = 8時間、**WEB SET OPTION**コマンドで変更可能)
- **WEB CLOSE SESSION**コマンドが呼び出された場合

このデータベースメソッドが呼び出された時点で、セッションのコンテキスト (プロセス変数の値やカレントセクション) は有効です。そのセッションに関連するデータ (変数の値やセクション) を退避し、後で同じcookie値でリクエストを受信したときにそれらを再利用することができます。

注: (複数のプロセスを生成可能な)4D Mobileセッションのコンテキストにおいて、**On Web Close Process データベースメソッド**は閉じられる各Webプロセス毎に呼び出されるので、4D Mobileセッションプロセス中に生成された全てのタイプのデータ(変数、セクション、等)を保存する事ができます。

On Web Close Process データベースメソッドの例題は[Webセッション管理](#)を参照してください。

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4D Mobile 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4D Mobile アクセスの保護:** REST経由の4D Mobile 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 パスワードを使用した自動コントロール](#) を参照のこと)
 - [On 4D Mobile Authentication database method](#) データベースメソッド
- **4D 側で4D Mobile サーバーへの公開を管理:** 4D Mobile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。
この設定は外部モデルの .js 設定ファイル内にて設定する事ができます([外部ファイルの変更](#)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。
計算フィールドは、外部モデルの .js 設定ファイル内に追加する事ができます([外部ファイルの変更](#) を参照して下さい)。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。
 - **拡張されたdatastore class と[restricting queries](#)を組み合わせる:** この強力な組み合わせを使用すれば、公開されている属性を管理するだけでなく、その属性が表示できるデータまで管理する事ができます。*datastore class* を拡張する事とは、つまり計算属性を追加したり既存の属性を削除したりすることによって変更可能なコピー(継承されたクラス)を作成するという事です。これに、*restricting query* を組み合わせることもできます。この場合、継承されたクラスのデータへのアクセスは全てクエリを起動し、それにより条件に合致したレコードをのみを返します。この原理により、データをWakanda Serverに接続しているユーザーと関連付ける事ができます。例えば、売り上げのデータベースにおいて、カレントのセールスパーソンに関連した顧客のみを返すクエリ、などが考えられます。もちろん、Webクライアントがアクセスできるのは継承されたクラスのみです。
拡張されたdatastore classes と *restricting queries* は、外部モデルの .js 設定ファイル内に作成・追加することができます([外部ファイルの変更](#) を参照して下さい)。詳細な情報に関してはWakanda ドキュメントの [Programming Restricting Queries](#) を参照して下さい。

注: 4D Mobile にて *restricting queries* を使用するためには、以下のシステム要件が最低限必要になります:

- 4D、4D Server の**v14.1**
- Wakanda Enterprise Server **v8**