

MY FIRST APPLICATION WITH 4D FOR FLEX

In this tutorial, we will show how easy it is to query records from a 4D v11 SQL Server and display them in a Flex DataGrid using 4D for Flex.

REQUIREMENTS

To complete this tutorial you will need to install the following software and files:

FLEX BUILDER 3 (SDK INCLUDED)

- Try: http://www.adobe.com/go/devcenter_flex_try
- Buy: http://www.adobe.com/go/devcenter_flex_buy

4D V11 SQL V11

- Try: <http://www.4d.com/products/downloads.html>
- Buy: <http://www.4d.com/purchase.html>

SAMPLE FILES:

Flex source : myFirstFourdFlexApp folder
4D database : 4DforFlex.4DB in 4DforFlex_demo folder

PREREQUISITE KNOWLEDGE

To benefit most from this tutorial, it is best if you:

- Have already programmed a few Flex applications
- Can distinguish between MXML and ActionScript 3.0 code at a glance
- Have basic knowledge of SQL

INTRODUCTION

Please note that we will run this code as a Flex app, using Flash Player for runtime, but it will work quite similarly as a desktop application using Adobe Integrated Runtime (AIR).

See the Adobe site if you are not familiar with AIR: <http://www.adobe.com/products/air/>

THE EXAMPLE DATABASE

For this example, we want to display the players of a baseball team.
We will search for players in a Player table which is related to a Team table:

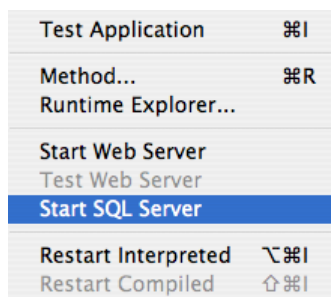


The image shows two screenshots of database tables. The left screenshot is titled 'Player' and has a pink background. It lists fields: ID (2³²), ID_Position (2³²), Name (A), First_Name (A), Bats (A), Throws (A), Height (A), Weight (0.5), ID_Country (2³²), Age (2¹⁶), CL (traffic light icon), DL (traffic light icon), Number (2¹⁶), ID_Current_Team (2³²), and Photo (camera icon). The right screenshot is titled 'Team' and has a cyan background. It lists fields: ID (2³²), ID_Pays (2³²), ID_Division (2³²), Name (A), Nickname (A), Team_logo (camera icon), Cap_Insignia (camera icon), Uniform (camera icon), Web_Site (T), Established (2¹⁶), ID_City (2³²), Color_Font (2³²), Color_Bkg (2³²), and Acronym (A).

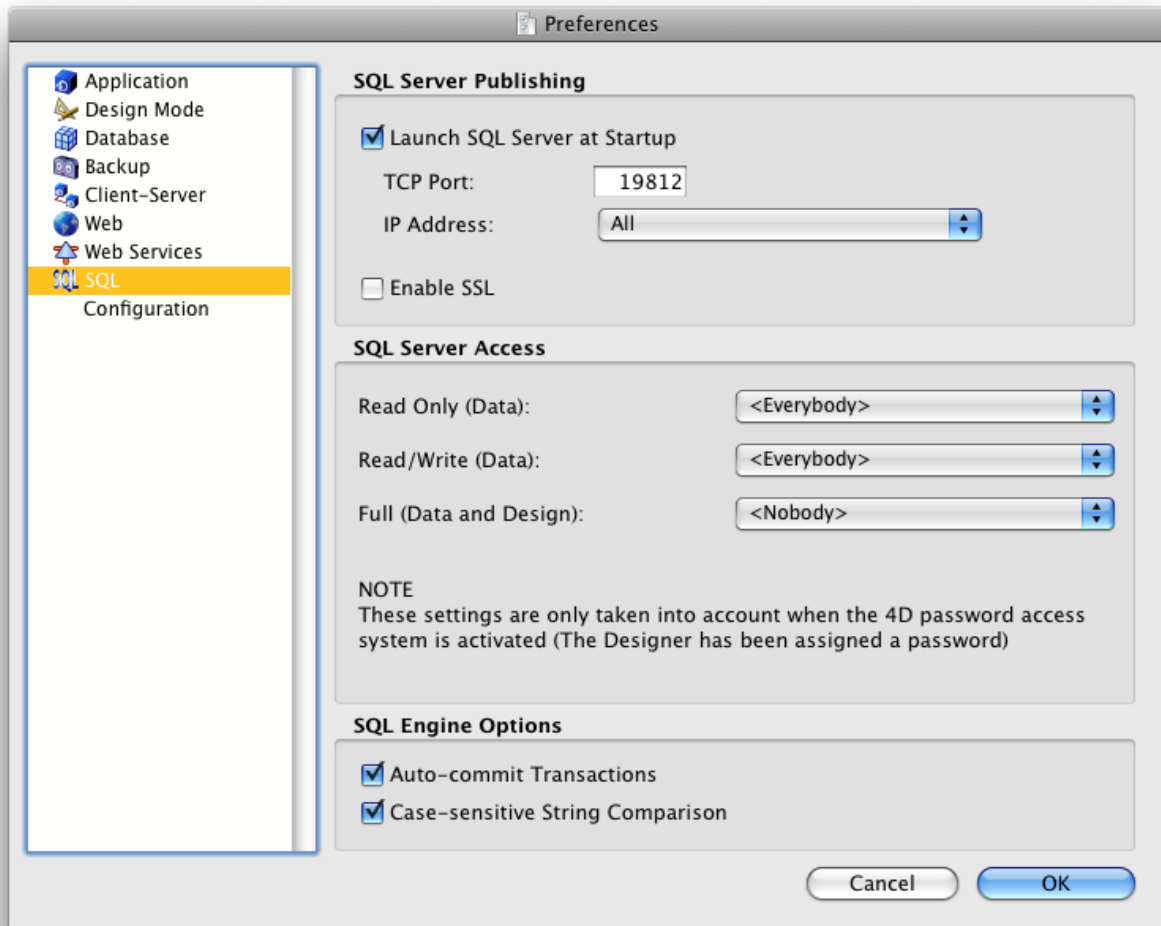
The [Player.ID_Current_team] field contains the [Team]ID of the player's current team.
So, by searching for [Player.ID_Current_team]=[Team] ID, it is easy to find the players of the current team.

SETTING THE 4D SQL SERVER

SQL Server must be started in order to allow 4D for Flex to execute queries. We can start it each time we want to do a query:



Or we can set the preferences so that SQL Server is automatically launched at Startup:



SECURITY

Please see the 4D for Flex manual and the 4D documentation for security concerns. You must open the SQL Server TCP port on firewalls. Then, you need to save a flash policy file allowing connection to SQL Server port in a folder inside the DB package: Preferences/SQL/Flash. This file called `socketpolicy.xml` may contain something like that:

```
<cross-domain-policy>
  <allow-access-from domain="*" to-ports="19812"/>
</cross-domain-policy>
```

PROGRAMMING THE FLEX CLIENT

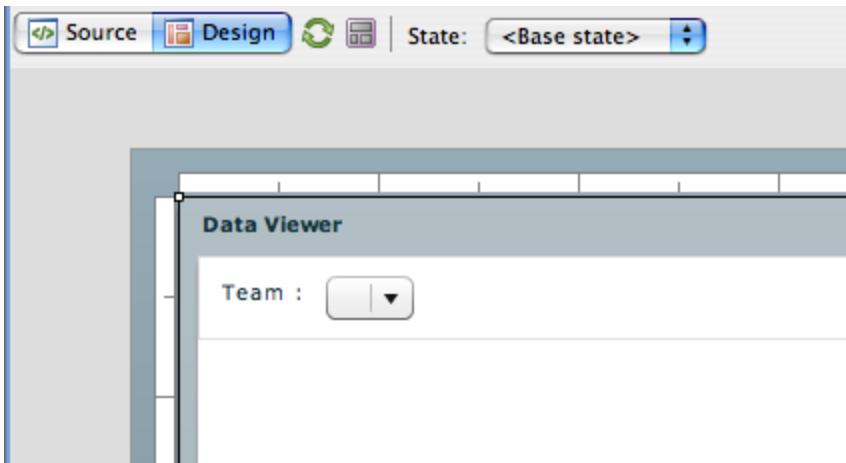
4 My First connection using 4D for Flex

No other action is required server-side. We can now program the Flex Client using Adobe Flex Builder 3. Please refer to the Adobe site for more information on Flex Builder: <http://www.adobe.com/products/flex/productinfo/overview/>

We will build our Flex Client in four steps. At the end of each step, we will have a working program that we can test and on which we can build the next step.

STEP 0: THE SKELETON

We start with a Flex skeleton containing the basic UI controls for our purpose:



A 'Team' combo-box will be populated with static data (we use the semi-dynamic XML template "teamsModel.xml" in 4D to get this data instantaneously) in order to list the teams. In a more evolved version, we will query 4D for this data too.

In later versions, when the user selects a team, a SQL query for the selected team will be sent to 4D and a DataGrid will list the players of this team.

You will find this bare skeleton in step 0.

We don't describe the Flex code for that since it is not at all using 4D for Flex. It's a very basic Flex app.

First let's populate the 'Team' combo-box.

STEP 1: THE TEAM COMBO-BOX

Let's execute a 4D method named 'flex_getTeams' which will paste the dataProvider needed by the combo-box into the clipboard:

```
<mx:Array>
<mx:Object label="Arizona Diamondbacks"/>
<mx:Object label="Atlanta Braves"/>
<mx:Object label="Baltimore Orioles"/>
(...)
```

We use a dynamic XML template named 'teamsModel.xml':

```
<mx:Array>
  <!--#4DLOOP [Team]-->
    <mx:Object label="<!--#4DVAR [Team]Name-->" />
  <!--#4DENDLOOP -->
</mx:Array>
```

This XML template is then parsed by the **PROCESS HTML TAGS** command, in order to populate the data according to the current [Team] selection:

```
ALL RECORDS ([Team])
ORDER BY ([Team]; [Team]Name;>)

C_TEXT($_vt_webFolder)
$_vt_webFolder:=Get 4D folder(HTML Root Folder) `v11.2 new selector
If (Test path name($_vt_webFolder)=Is a directory )

  C_TEXT($_vt_model)
  $_vt_model:=$_vt_webFolder+"teamsModel.xml"

  If (Test path name($_vt_model)=Is a document )
    C_BLOB($_vx_teamsModel)
    DOCUMENT TO BLOB($_vt_model;$_vx_teamsModel)

    C_TEXT($_vt_teams)
    PROCESS HTML TAGS($_vx_teamsModel;$_vt_teams)

    SET TEXT TO PASTEBOARD($_vt_teams)
    ALERT("The teams are pasted in clipboard")

  End if

End if
```

See the 4D documentation for more information:

<http://www.4d.fr/documentation/4DdocV11/CMU/CMU00816.HTM>

<http://www.4d.fr/documentation/4DdocV11/CMU/CMU02070.HTM>

Then, let's assign the dataProvider of the Combo box in the Flex code. Select the combo-box in Design mode, switch to 'Source' displaying, you should see this:

```
<!-- combobox-->
:mx:Label text="Team :"/>
  <mx:ComboBox
    id="comboBox"
    change="select(event.target.text)"
  >

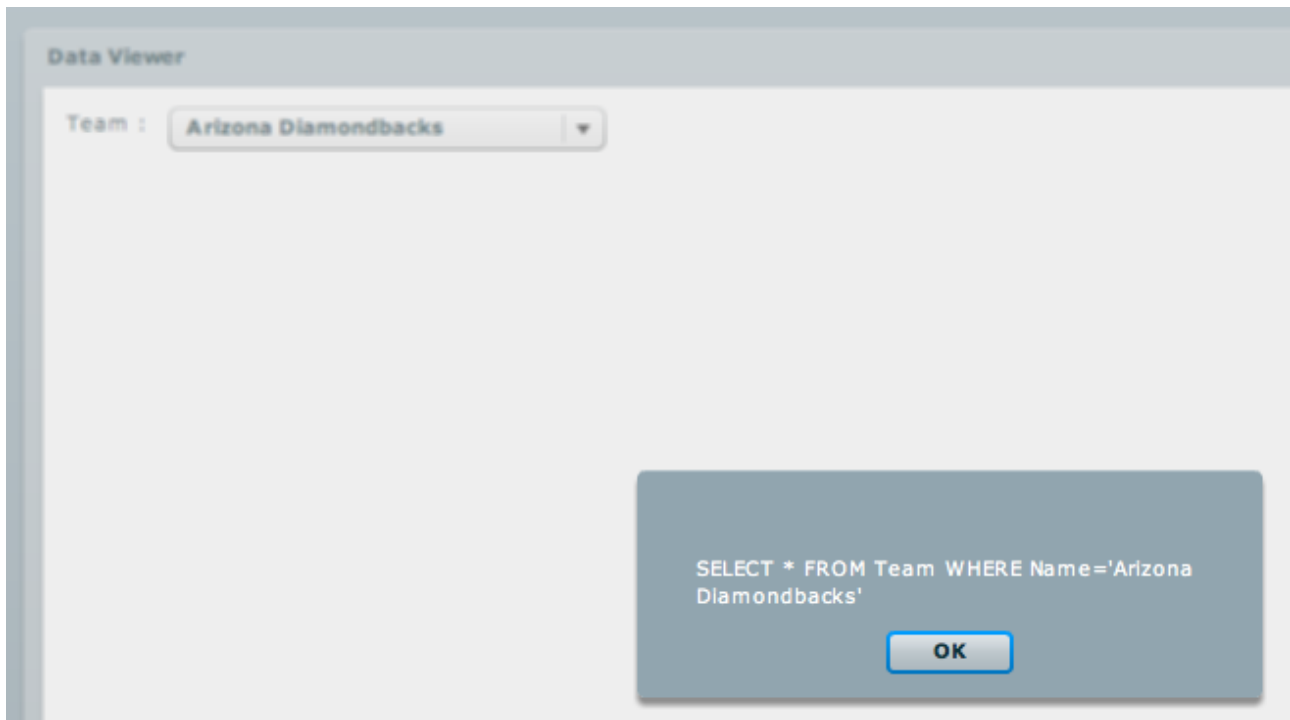
    <mx:dataProvider>
      <!-- combo dataProvider-->
    </mx:dataProvider>

  </mx:ComboBox>
```

So, let's just paste the text from the pasteboard below the <!-- combo dataProvider --> comment:

```
76 <mx:dataProvider>
77   <!-- combo dataProvider-->
78   <mx:Array>
79     <mx:Object label="Arizona Diamondbacks"/>
80     <mx:Object label="Atlanta Braves"/>
81     <mx:Object label="Baltimore Orioles"/>
82     <mx:Object label="Boston Red Sox"/>
83     <mx:Object label="Chicago Cubs"/>
84     <mx:Object label="Chicago White Sox"/>
85     <mx:Object label="Cincinnati Reds"/>
86     <mx:Object label="Cleveland Indians"/>
87     <mx:Object label="Colorado Rockies"/>
88     <mx:Object label="Detroit Tigers"/>
89     <mx:Object label="Florida Marlins"/>
    etc.
```

Save the source, click on the green 'Run' button and see the result:



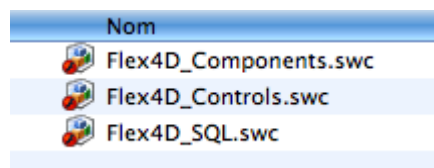
When an item is selected in the combo-box, an alert is displayed, showing the SQL query to be executed. Since the first item is selected at startup, the alert is displayed for the first team.

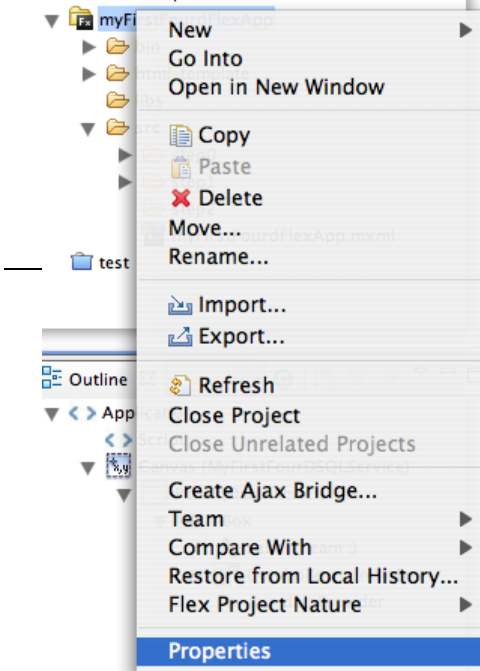
Fine, let's see how we can execute the query...

STEP 2: EXECUTE THE QUERY

Now, we will use 4D for Flex. Let's see how it is provided. We have three components:

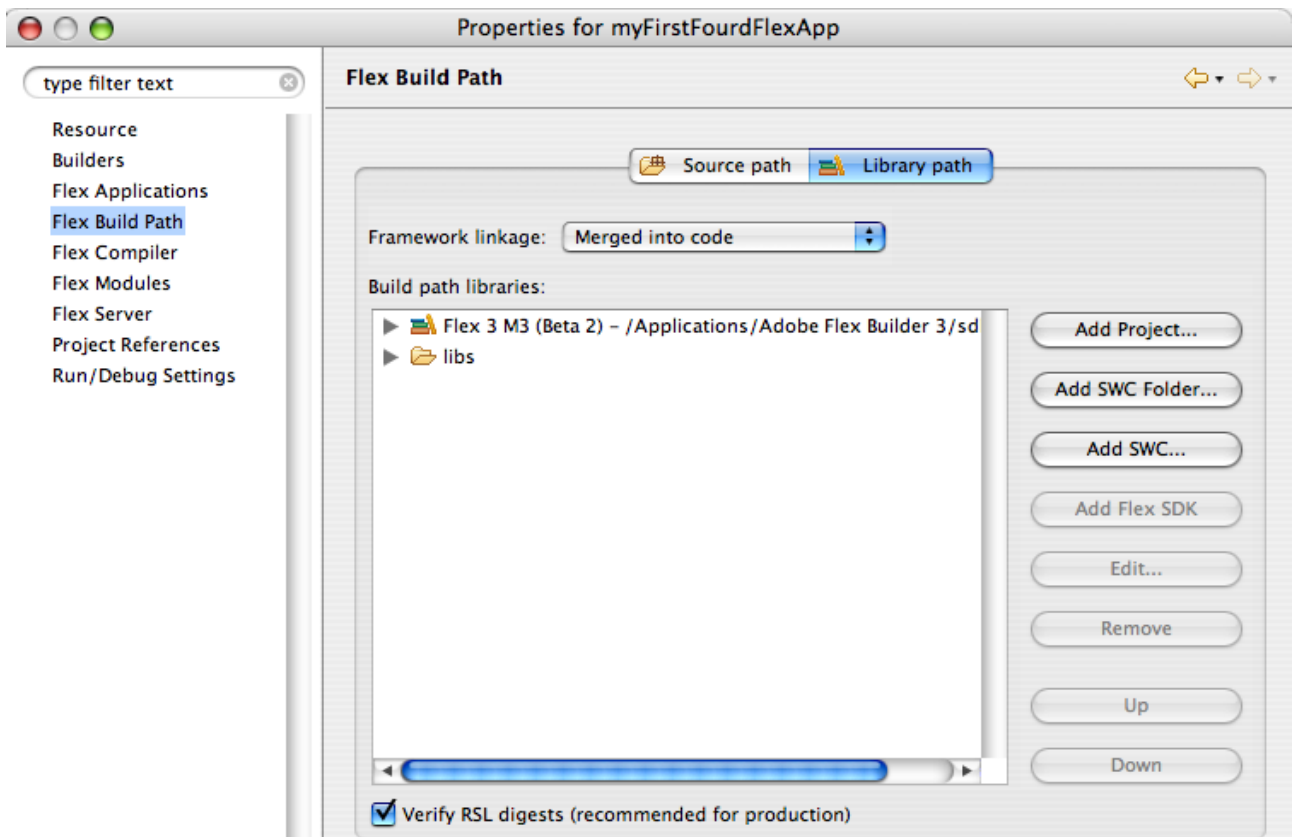
- Flex4D_Components: contains utilities like the Authentication component and renderers;
- Flex4D_Controls: contains enhanced UI controls such as a custom DataGrid;
- Flex4D_SQL: contains all we need to connect to 4D v11 SQL Server and execute SQL statements.





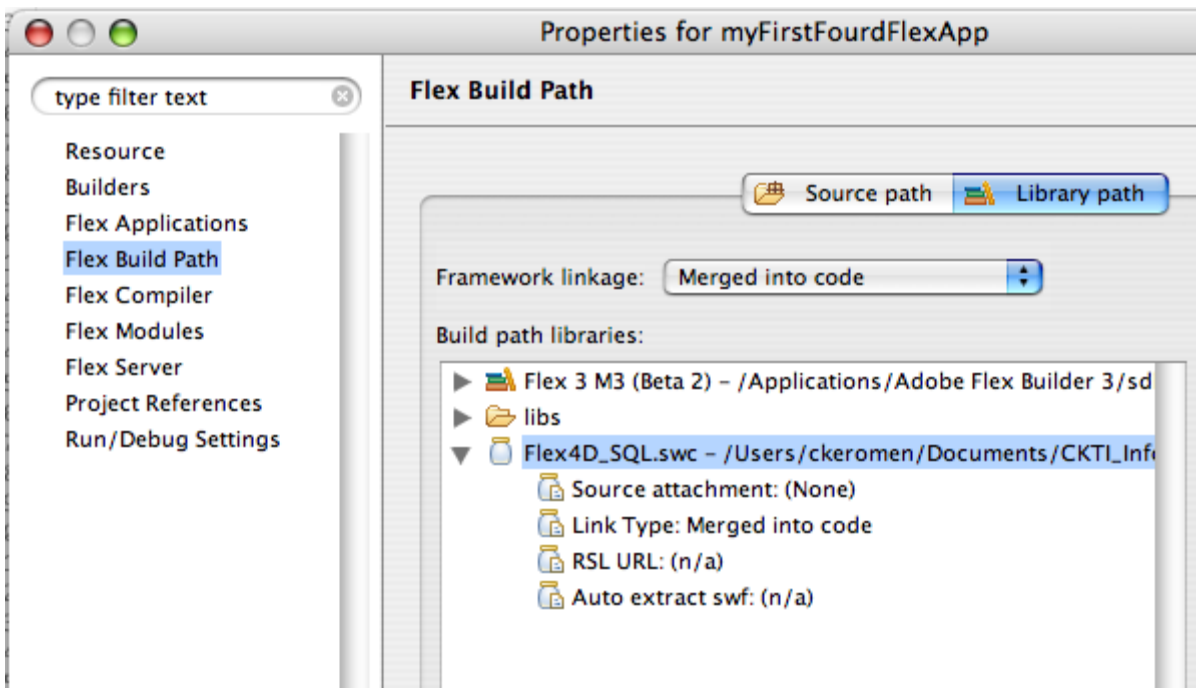
A SWC file is a Flex component. We need to declare that our project will use Flex4D_SQL.swc and Flex4D_Controls.swc.

First, we must declare in our Flex project where the components are on our disk. Select the project properties, then



Flex Build path and go to the 'Library path' tab:

Click on 'Add SWC...' and select the location of the desired component.



Let's do the same for Flex4D_Controls.swc.

Ok, our project is now ready to use 4D for Flex.

In order to execute a SQL query, we must declare a SQLService in MXML: this component will handle the communication between Flex and 4D SQL Server.

Switch back to the code.

First, we must declare the correct namespace for our component: <http://www.4d.com/2007/mxml>

```
<mx:Application
    xmlns:mx="http://www.adobe.com/2006/mxml"
    creationComplete="select (comboBox.selectedLabel)"
    xmlns:fourD="http://www.4d.com/2007/mxml"
>
```

Then type the following code just below the `<!-- 4DFlexLib SQLDataService-->` comment; auto-completion should activate as follows:

```

<!-- 4DFlexLib SQLDataService-->
<fourD:SQLService
  id <> fourD:Button
  hos <> fourD:DataGrid
  use <> fourD:NavigationButtonGroup
  aut <> fourD:SQLService

```

Select `fourD:SQLService`. Then auto-completion helps you again to set the properties and the event handlers needed by the component:

```

<!-- 4DFlexLib SQLDataService-->
<fourD:SQLService
  activate
  autoConnect
  connect
  deactivate
  disconnect
  fault
  host
  id
  port
  preferredImageTypes
</mx:

```

Select and set the different items to obtain the following statement:

```

<fourD:SQLService
    id = "fourDSQLService"
    host = ""
    userName = "Administrator"
    autoConnect="true"

    result = "resultHandler(event)"
    fault = "faultHandler(event)"
/>

```

Let's comment some properties:

- **host** is the URL where 4D SQL Server can be requested; when a blank value is provided, it will use the domain from which the SWF is served or localhost by default. When using AIR, you must always declare a host.

- **userName** is the login name for authentication (a `userPassword` will be necessary as well in real life); it must match an existing 4D User or be handled in on SQL Authentication Database method (see V11.2 manual).
- **autoConnect** is set to `True` to tell the component to try connecting automatically to the Server when created in memory.

We also declare two event handlers: the first, `resultHandler`, is called when a result is received from the Server and the second, `faultHandler`, is called when something goes wrong.

We must now add these handlers to the `<mx:Script>` tag. Search for the `// Event handlers` comment and add the following ActionScript code:

```
/**
the call to SQLService returned a result
*/
private function resultHandler(event:ResultEvent):void
{

}

/**
the call to SQLService returned an error
*/
private function faultHandler(event:FaultEvent):void
{
    Alert.show("Error #" + event.fault.faultCode + ": " +
event.fault.faultString);
}
```

Then save... and you get syntax errors:

```
--
34     l'appel au SQLService a retourné un résultat
35     */
36     private function resultHandler(event:ResultEvent):void
37     {
38
39     }
40
41     /**
42     l'appel au SQLService a retourné une erreur
43     */
44     private function faultHandler(event:FaultEvent):void
45     {
46         Alert.show("Error #" + event.fault.faultCode + ": " + event.fault.faultString);
47     }
48
```

Why?

Because ResultEvent and FaultEvent are unknown types of events. So we must tell the compiler about these events.

Add the following import statement below the '/// packages mx.rpc.events' string and save.

```
/// packages mx.rpc.events
import mx.rpc.events.FaultEvent;
import mx.rpc.events.ResultEvent;
```

Now we can add an alert in resultHandler:

```
Alert.show(event.toString());
```

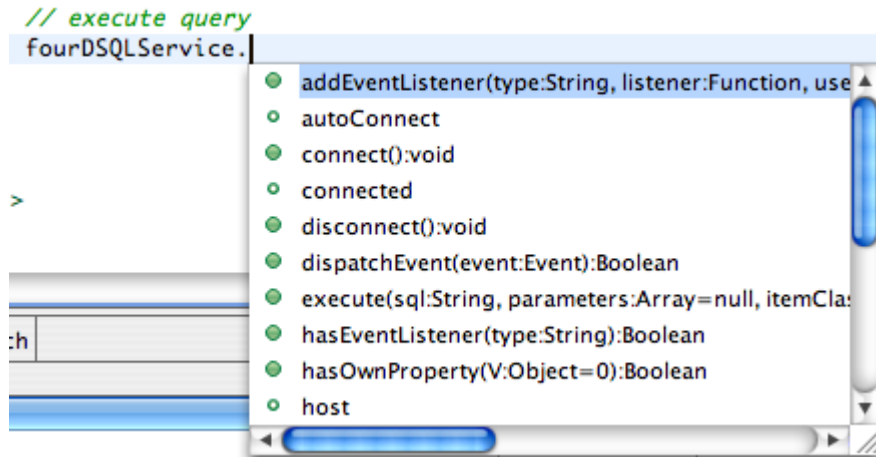
This alert will be displayed when 4D SQL Server sends a reply. But first we must send a query!

So, let's replace the code showing an Alert in select() function. Remember, this is the code triggered when selecting an item in the combo-box.

We receive the selected team as a parameter. So, this very basic SQL query will load the corresponding team:

```
"SELECT * FROM Team WHERE Name='" + item + "'"
```

Type the ID of the SQLService followed by a period. The auto-completion function will show you properties and functions for the Service.



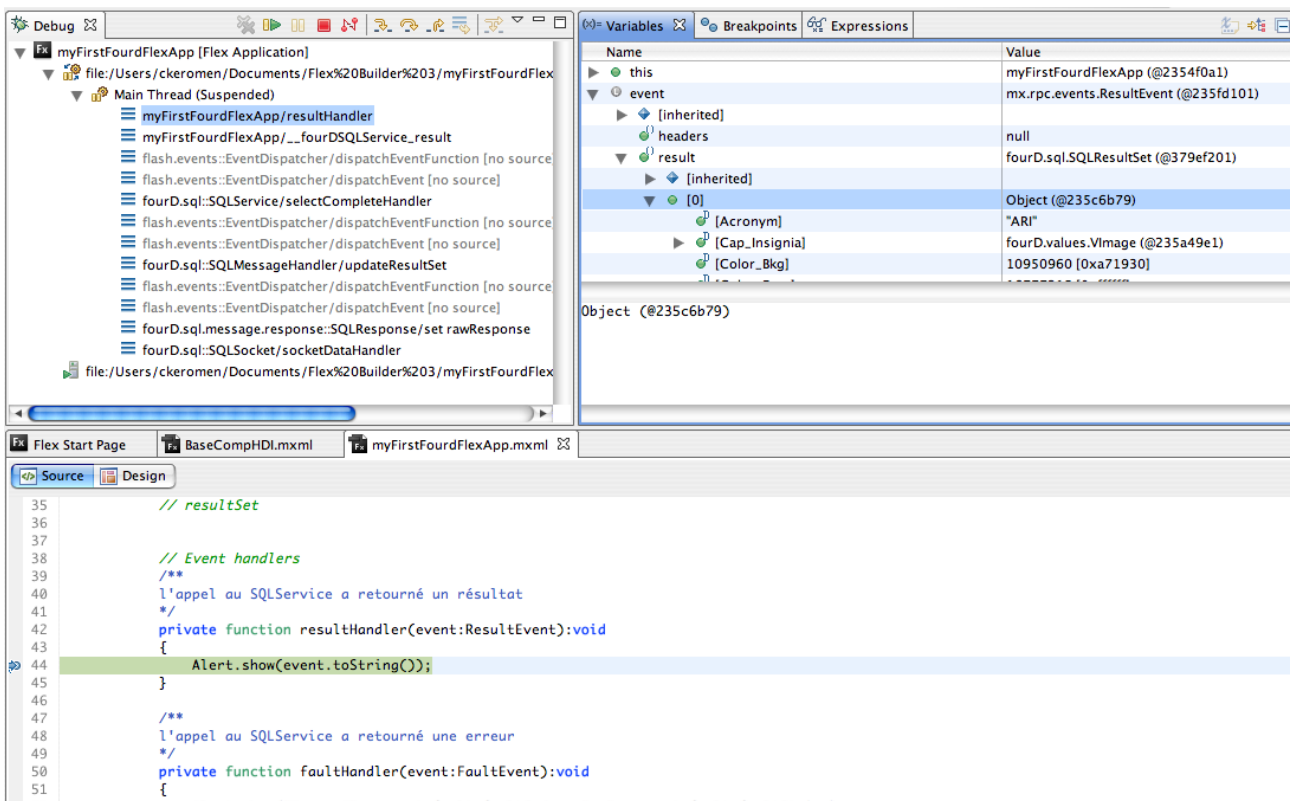
Select the `execute()` function and pass the SQL string as the only parameter; the others are not mandatory. You should get the following code:

```
private function select(item:String):void
{
    var sql:String = "SELECT * FROM Team WHERE Name='" + item +
        "'";

    // execute query
    fourDSQLService.execute(sql);
}
```

Save, run and enjoy: you just executed your first SQL query from Flex to 4D SQL Server!

Here, in debug mode, you can see some details of the data returned by 4D SQL Server:



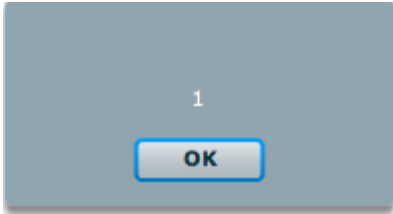
If you explore the event.result object, you can see its type (fourD.ResultSet) and its properties. Among them, we can choose event.result.nbRecords and display it in our Alert. Let's modify the code:

```

private function resultHandler(event:ResultEvent):void
{
    Alert.show(event.result.nbRecords);
}

```

Save, run, and you should get this:



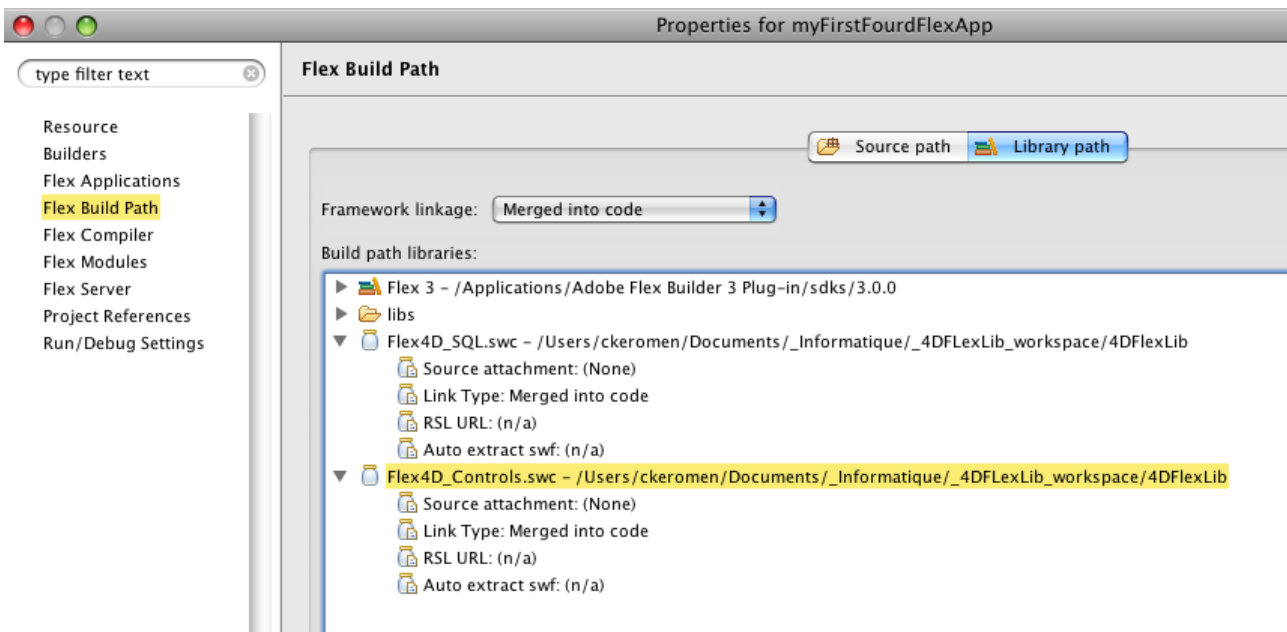
In step 3, we'll display the results in a DataGrid.

STEP 3: DISPLAYING RESULTS IN A DATAGRID

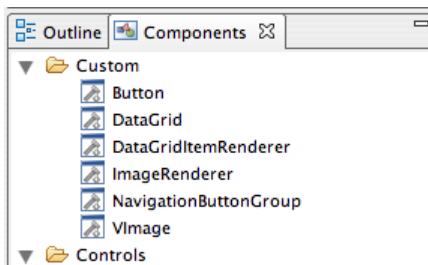
We will then use another component of the 4D for Flex, the Flex4D_Controls.swc, which contains enhanced UI controls (DataGrid, navigation buttons, etc.) These controls inherit from Flex controls and benefit from the Flex4D_SQL component while adding some high-level automatic functioning.

For instance, the DataGrid automatically uses an ImageRenderer to display images loaded from the SQL Server.

So, modify the build path to add a reference to this second component.



Switching back to Design mode, you'll see that you now have Custom Components:



Drag-and-drop the Custom DataGrid on stage, switch to Source mode and set the properties as follows:

```
id="item_dg"  
width="100%"  
height="95%"
```

Ok, the DataGrid is declared, but we need to set its dataProvider property in order to display the result of our SQL query.

We saw in a previous step that the result event fired contains a result object of the SQLResultSet type. This object inherits from an ArrayCollection and can be used directly as a dataProvider for the DataGrid.

So, let's just declare a new instance variable `_resultSet` of the SQLResultSet type and let's make it Bindable so we can bind it to the DataGrid dataProvider.

Find the `// resultSet` comment in the Script part and add the following:

```
[Bindable]  
private var _resultSet:SQLResultSet;
```

We must add an import statement for the SQLResultSet type which is part of 4D for Flex.

```
// packages fourD  
import fourD.sql.SQLResultSet;
```

Ok, now we've got a bindable variable that we can bind to the dataProvider property of the DataGrid.

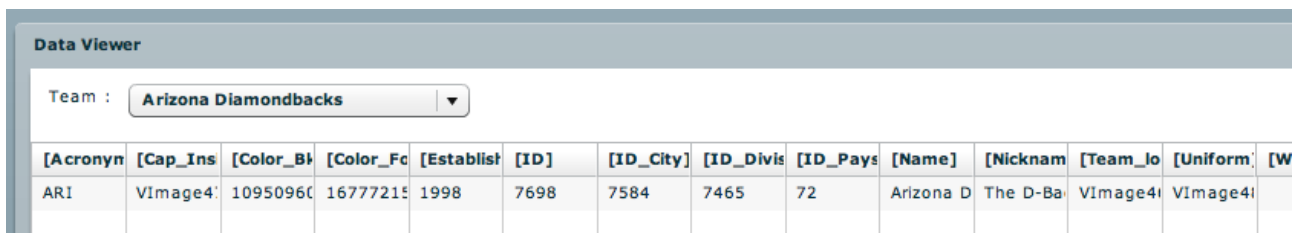
```
dataProvider="{_resultSet}"
```

4D for Flex 1.0

To complete the process, we must fill `_resultSet` with the result sent by the Server. That will be done in the `resultHandler()` function. Let's type:

```
_resultSet = event.result as SQLResultSet;
```

Save, run... enjoy.



The screenshot shows a 'Data Viewer' window with a dropdown menu set to 'Arizona Diamondbacks'. Below the menu is a table with 13 columns and one row of data. The columns are labeled with field names in brackets, and the row contains the corresponding values for the Arizona Diamondbacks team.

[Acronym]	[Cap_Ins]	[Color_B]	[Color_Fc]	[Establish]	[ID]	[ID_City]	[ID_Divis]	[ID_Pays]	[Name]	[Nicknam]	[Team_Jo]	[Uniform]	[W
ARI	VImage4	10950960	16777215	1998	7698	7584	7465	72	Arizona D	The D-Ba	VImage4	VImage4	

The DataGrid is correctly populated by the data retrieved from 4D SQL Server.

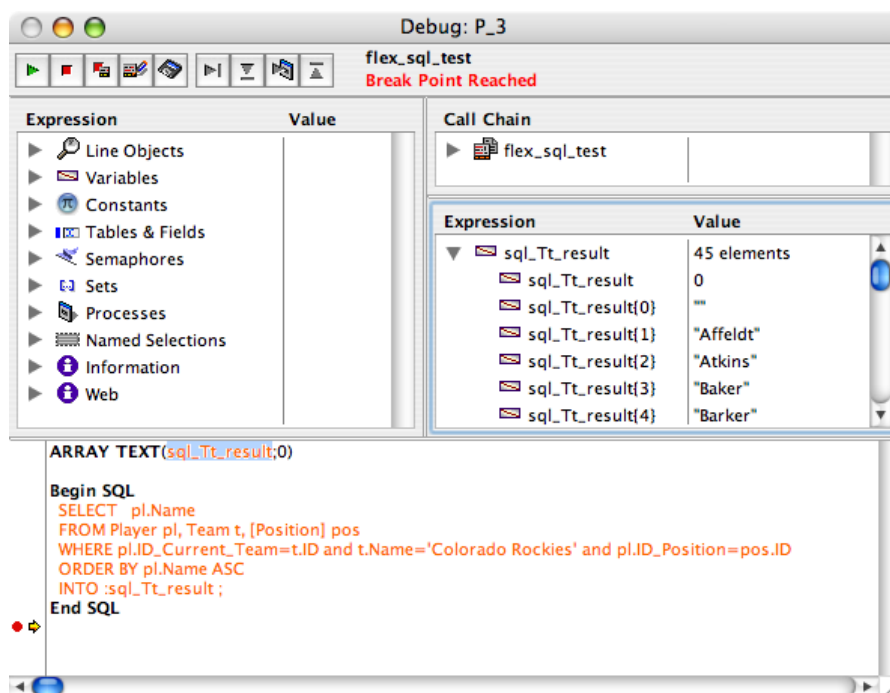
But, what if we want to see not the Team record, but the Players of the selected team?

Let's accomplish the final step by simply changing the SQL query.

STEP 4: QUERYING THE PLAYERS

We only need to change the SQL query in our Flex code!

If we're not sure of our SQL statement, we can test it directly in 4D:



So let's change the code in our `select()` function:

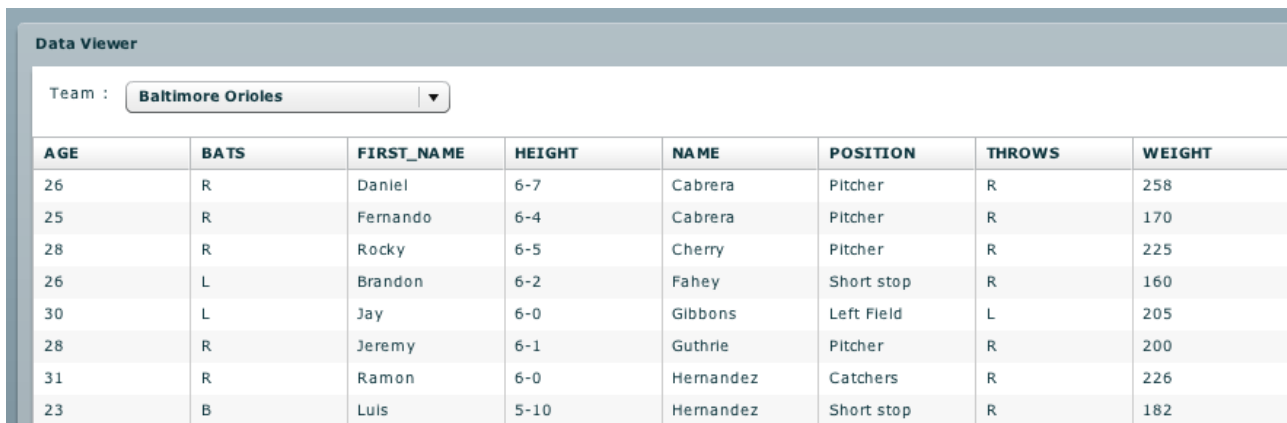
```

var sql:String = "SELECT " +
    "pl.First_Name, " +
    "pl.Name, " +
    "pl.Age, " +
    "pos.Name as [POSITION], " +
    "pl.Bats, " +
    "pl.Throws, " +
    "pl.Height, " +
    "pl.Weight " +
    "FROM " +
    "Player pl, " +
    "Team t, " +
    "[Position] pos " +

```

```
"WHERE " +  
    "pl.ID_Current_Team=t.ID " +  
    "and t.Name='" + item + "' " +  
    "and pl.ID_Position=pos.ID " +  
"ORDER BY " +  
    "pl.Name ASC";
```

Save, run... enjoy:



The screenshot shows a 'Data Viewer' window with a dropdown menu set to 'Baltimore Orioles'. Below the menu is a table with 8 columns: AGE, BATS, FIRST_NAME, HEIGHT, NAME, POSITION, THROWS, and WEIGHT. The table contains 9 rows of player data.

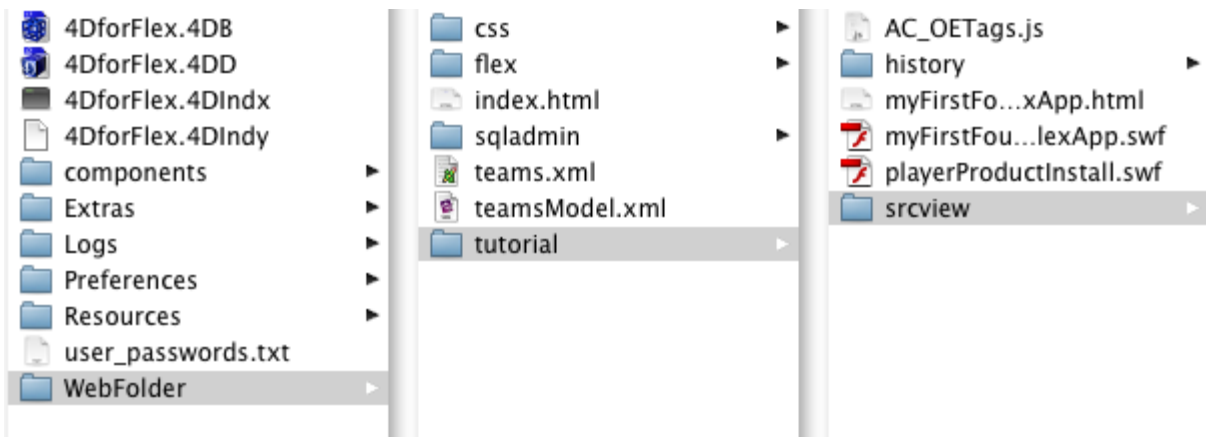
AGE	BATS	FIRST_NAME	HEIGHT	NAME	POSITION	THROWS	WEIGHT
26	R	Daniel	6-7	Cabrera	Pitcher	R	258
25	R	Fernando	6-4	Cabrera	Pitcher	R	170
28	R	Rocky	6-5	Cherry	Pitcher	R	225
26	L	Brandon	6-2	Fahey	Short stop	R	160
30	L	Jay	6-0	Gibbons	Left Field	L	205
28	R	Jeremy	6-1	Guthrie	Pitcher	R	200
31	R	Ramon	6-0	Hernandez	Catchers	R	226
23	B	Luis	5-10	Hernandez	Short stop	R	182

Since everything is working correctly, how do we now deploy this application?

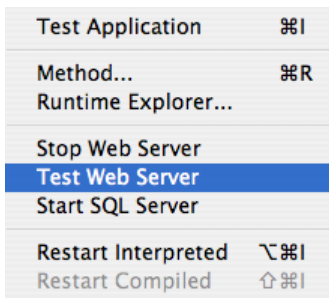
FINAL STEP: DEPLOYING THE FLEX APPLICATION

Just create a 'flex' folder in our database WebFolder.

In this folder, copy the contents of the bin folder of our Flex project to a tutorial folder in 4D WebFolder:



Check that 4D Web Server is running:



Then ask for the following URL in your browser:

<http://localhost:8080/tutorial/myFirstFourFlexApp.html>

The HTML page, myFirstFourFlexApp.html, containing the SWF, has automatically been generated by Flex Builder.

SUMMARY

In this tutorial, you've learned how simple it is to link your Flex project to 4D for Flex components, declare and set a SQLService component, then bind data returned by 4D SQL Server to a Flex DataGrid.