



AppBuilder 2.0

bhutten@pelicaneng.com

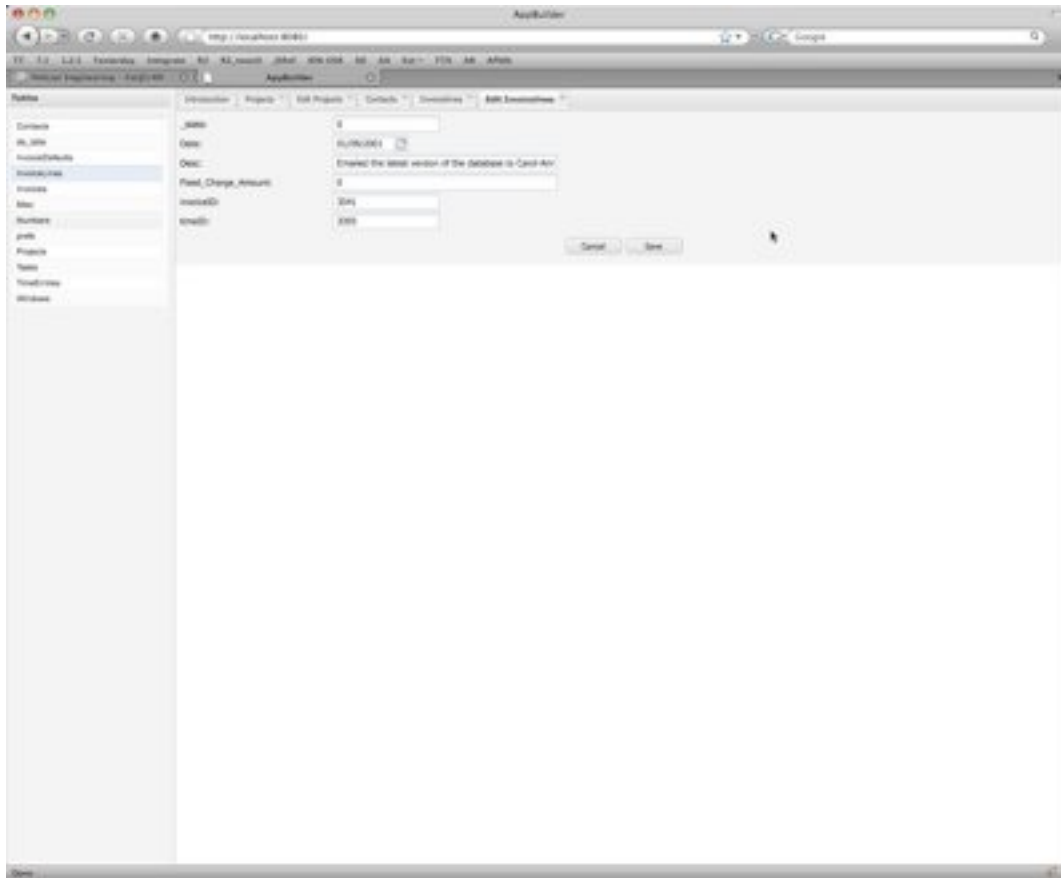
May 18/2010

AppBuilder is a 4D database/component that quickly allows you to add an ExtJS web interface – with full read/write/modify/delete capabilities – to your 4D v11 databases. Unlike some similar tools, AppBuilder is a code generator – it generates all the required ExtJS classes for the application, which you can then customize as desired.

Example grid form generated by Appbuilder:

id	id2	id3	id4	Address	Author	City	County	Date Created	Date Modified	Default	Status	Default
1	1	1	1	100 Main St.	100	100	100	100	100	100	All work is off	100
2	2	2	2	200 Main St.	200	200	200	200	200	200	All work is off	200
3	3	3	3	300 Main St.	300	300	300	300	300	300	All work is off	300
4	4	4	4	400 Main St.	400	400	400	400	400	400	All work is off	400
5	5	5	5	500 Main St.	500	500	500	500	500	500	All work is off	500
6	6	6	6	600 Main St.	600	600	600	600	600	600	All work is off	600
7	7	7	7	700 Main St.	700	700	700	700	700	700	All work is off	700
8	8	8	8	800 Main St.	800	800	800	800	800	800	All work is off	800
9	9	9	9	900 Main St.	900	900	900	900	900	900	All work is off	900
10	10	10	10	1000 Main St.	1000	1000	1000	1000	1000	1000	All work is off	1000
11	11	11	11	1100 Main St.	1100	1100	1100	1100	1100	1100	All work is off	1100
12	12	12	12	1200 Main St.	1200	1200	1200	1200	1200	1200	All work is off	1200
13	13	13	13	1300 Main St.	1300	1300	1300	1300	1300	1300	All work is off	1300
14	14	14	14	1400 Main St.	1400	1400	1400	1400	1400	1400	All work is off	1400
15	15	15	15	1500 Main St.	1500	1500	1500	1500	1500	1500	All work is off	1500

Example input form generated by AppBuilder:



The screenshot shows the AppBuilder application window. On the left is a sidebar with a tree view containing the following items: Context, No Site, AccountInfo, **Entities**, Events, Roles, Business, Profile, Projects, Roles, Roles, Roles, Roles. The 'Entities' item is selected. The main area displays the 'Properties' tab for a selected entity. The properties are as follows:

Property	Value
Name	Entity
Class	Entity
Desc	Entity: The latest version of the database is Card-App
Field_Change_Amount	0
Amount	0
Amount	0













At the bottom right of the properties section are 'Cancel' and 'Save' buttons.

Requirements

AppBuilder requires that each table in your 4D v11 database have a primary key field named "ID", which must be LONGINT.

Installation – Step 1

When you unzipped the AppBuilder archive you should have ended up with a folder with the following items in it:














Name	Size
 AB.4DD	201 KB
 AB_2-0.4DB	659 KB
▼  AB_Templates	--
 HTML_INDEX.txt	4 KB
 JS_GRID_TEMPLATE.txt	8 KB
 JS_INPUT_TEMPLATE.txt	4 KB
 JS_PALETTEPANEL.txt	4 KB
▼  Components	--
 JSON_1-0-13.4dbase	1.3 MB
▼  WebFolder	--
▶  images	--
 index.js	4 KB

Copy “AB_2-0.4DB” and “JSON_1-0-13.4dbase” into the “Components” folder of your host database.

Copy the “AB_Templates” folder into the same folder as your host database.

Copy the “images” folder and “index.js” file from the WebFolder into the WebFolder of your host database.

Your host db folder should now look like this:

Name	Size
 Test Host.4DD	201 KB
 Test Host.4DB	463 KB
▼  AB_Templates	--
 HTML_INDEX.txt	4 KB
 JS_GRID_TEMPLATE.txt	8 KB
 JS_INPUT_TEMPLATE.txt	4 KB
 JS_PALETTEPANEL.txt	4 KB
▼  Components	--
 ab_2-0.4dbase	664 KB
 JSON_1-0-13.4dbase	1.3 MB
▼  WebFolder	--
▶  images	--
 index.js	4 KB

Step 2

Open your host DB. In the "On Web Connection" method of your database you need to initialize the JSON component and add a call to the "ab_onWebConnection" method:

```
C_TEXT($1;$2;$3;$4;$5)

json_init

If (Not(ab_onWebConnection ($1;$2;$3;$4;$5;$6)))
  `Your code here
End if
```

Also make certain that your host database has web serving turned on.

Step 3

Execute the “ab_build” method. AppBuilder will step through each table in your host database and build a set of ExtJS classes for that table. In our example “Test Host” database, there are two tables:



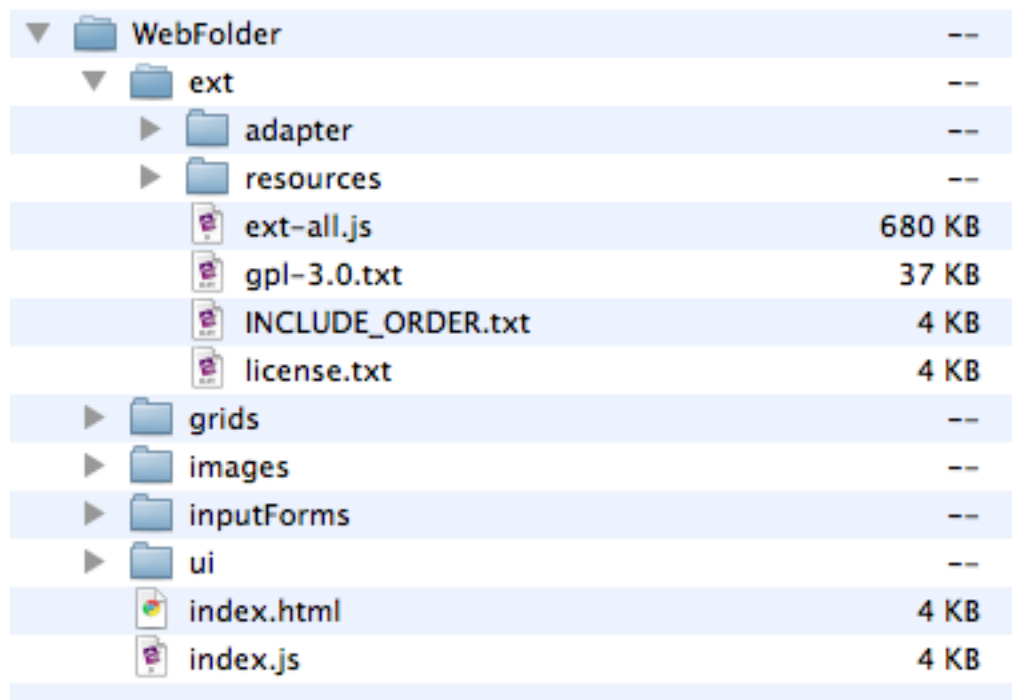
These files will all be written into the “WebFolder” of your database. In the case of our example database, our WebFolder now looks like this:

▼	WebFolder	--
▼	grids	--
	Companies_Grid.js	8 KB
	People_Grid.js	8 KB
▶	images	--
▼	inputForms	--
	Companies_Input.js	4 KB
	People_Input.js	4 KB
▼	ui	--
	PalettePanel.js	4 KB
	index.html	4 KB
	index.js	4 KB

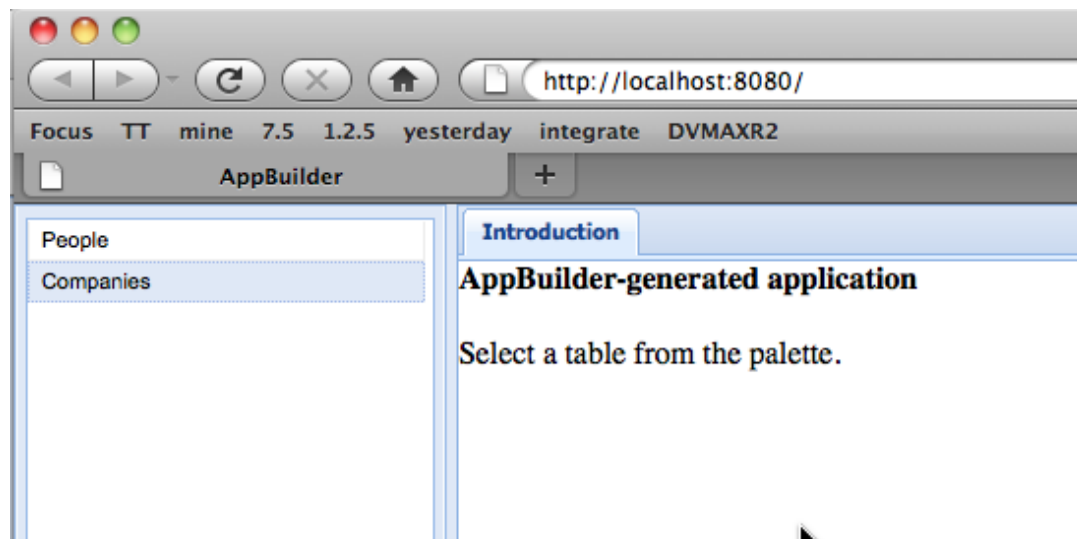
AppBuilder has generated the files in the “grids”, “inputForms”, and “ui” folders, as well as the “index.html” file.

Step 4

Now you need to take the ExtJS folder (from extjs.com), copy it into your webfolder, and rename it "ext", ie:



You can now access your generated application:



Note: The ExtJS classes for grids and input forms use the "pre-configured class" pattern. See Writing a Big Application in ExtJS (Parts 1–3) for more information:

http://extjs.com/learn/Tutorial:Writing_a_Big_Application_in_Ext

<http://blog.extjs.eu/know-how/writing-a-big-application-in-ext-part-2/>

<http://blog.extjs.eu/know-how/writing-a-big-application-in-ext-part-3/>