Gradebook: a simple data-driven app using CustomListViews, a HorizontalScrollView, B4Xpages and B4Xdialogs.

In this tutorial you can learn how to write a data-driven app. This means that the app displays and uses information coming from a (SQLite) database.

User Interface

In the B4XMainPage the classes with the studentlists are displayed. For each class there is a CustomListView that contains the list of students from that class. You can scroll horizontally to the next or previous class.

The menu in the B4XMainPage contains entries to the other B4Xpages: class, course, enrollment, score, student and test. Each B4Xpage (except score) has a menu with an entry for the CSV import file dialog.

Database

There are no foreign key constraint violation checks when you delete a record from a table!
You can safely delete records from the score table or the note table.

When you start the app for the first time the database is initialized and the tables are created.

Import CSV files

When you start the app for the first time you have to provide the data. To facilitate this each B4Xpage (except the score page) has a menu entry CSV import.

In the dialog you can choose a file from the B4XComboBox. This combobox is filled with the filenames from the importfolder (Main storage – Android – data – b4a.gradebook – files – Import) that was created the first time at startup.

Unpack the "gradebook_csv_files.zip" file and copy the files to the importfolder on the device.

Restart the app and you will see the files.

Because of the dependencies between the database tables it is important to import the CSV files is the right order: classes.csv, students_4oit.csv, students_5it.csv, students_6it.csv, courses.csv, enrollments.csv and tests.csv.

Then you are ready to test the app.

In the B4XMainPage you can tap on a student and scroll through the information: student info, class info, enrollment info, test info (if available) and note info (if available).

In the score page you can choose a test and start giving the student a score on that test. You can give a short comment for that test result.

In the B4XMainPage you can long tap (ItemLongClick) on a student name to write a note for that student.

Source code

In the Main page of the project you can find the declarations of the 2 database class variables and the type variables for each database table. Each type variable has the same structure as the database table with the same name.

```
Sub Process_Globals
    Public ActionBarHomeClicked As Boolean
    Public db As database
    Public ds As dataset
    Type class(rowid As Int, shortname As String, longname As String, location As String)
    Type course(rowid As Int, name As String, description As String)
    Type enrollment(rowid As Int, period As String, course_id As Int, student_id As Int)
    Type note(rowid As Int, notetext As String, student_id As Int)
    Type score(rowid As Int, score As Int, comment As String, student_id As Int, test_id As Int)
    Type student(rowid As Int, firstname As String, lastname As String, birthdate As String, class_id As Int)
```

```
Type test(rowid As Int, date As String, time As String, description As String, maxscore As Int, testtype As String, course_id As Int)
End Sub
```

The rowid is also present in the type variable for easy access to it.

In the B4XMainPage the lists are filled with the “fill_clv” subroutine. This subroutine is called for each class in the B4XPage_appear subroutine. This means that the lists are refreshed when the user returns from another page or when the user enters a note for a student (long tap on a student).

When the app goes in the background (B4XPage_disappear) then the database is exported to the exportfolder as a form of backup.

The project contains 2 database classes: database and dataset. You can find the general database methods in the database class. The page specific database methods you can find in the dataset class.

The individual pages show a CustomListView, an add button and a dialog when the user taps on a listview item or taps on the add button. In the dialog the fields are presented in a CustomListView. When needed a combobox is presented. The user can save or delete a record.

The import_csv subroutine is used to fill the database tables with the information from the CSV files. The delimiter is a “;”.

The CustomListViews in this app use the clv.AddTextItem() method to display the information. You could change that and use a panel with labels, buttons, imageviews, …

At the beginning of each school year the teacher can modify the CSV files to his/her specific needs.

Remember to copy the database file from the export folder before you re-install the app to be used with the new CSV files!