Welcome to DataFlex 19.0 for Web

Data Access Worldwide is the creator of DataFlex. Since 1976, Data Access has been delivering tools for building database applications. DataFlex is also available in a Personal Edition, which is a free, fully functional edition for non-commercial, private use.

Download a copy of DataFlex 19.0 from www.download.com or www.dataaccess.com if you did not yet install the DataFlex Studio and install the product. Use the DataFlexWebAppCheck tool to test if you system is properly configured system for creating web applications.

The goal of this introduction is to show you the steps involved in building database applications with DataFlex. We will do so by using an example scenario. We will make a browser-based application, accessible via the Internet. This introduction will focus on Desktop style web applications. There is a different Quickstart guide available for DrillDown – Mobile/Touch style applications.

In a second step we will add reporting. Most of the functionality will be limited to what can be obtained by drag & drop features in DataFlex, but we included some small code fragments. Don't let it stop you from getting further into the underlying programming language and framework!

Let us start by introducing some concepts to you.

Object oriented

DataFlex is an object oriented programming language. This means that the product – out of the box – delivers classes for all common components. This class is the definition of how such a component looks and functions and what it exactly does once it is instantiated as an object in a written program. The advantage is that a lot of technical details are taken care of for you and you can concentrate on the real functionality of the application you want to develop.

Workspace

Before you start a web project, it is required to make a new environment on your PC, called a "workspace". A workspace is a set of folders in which the database, source-code and such are stored. A workspace can only have one web project. This project will be compiled into a program or 'executable'. The workspace and project information is viewed from within the DataFlex Studio through the Workspace Explorer. You can also see a workspace's directory structure in the Configure Workspace Properties dialog.

Themes

The Framework is highly CSS (Cascading Style Sheet) based for the layout. Switching – even live – to a different theme changes the visual look and feel of the application but requires no code changes. The framework installs five different themes with the Df_Web_Creme as the default theme for web application development. Take a look at one of the five themes in the WebOrder example. There is even a Windows alike theme available. In this introduction guide we won't go in CSS itself, that is more an expert level feature.
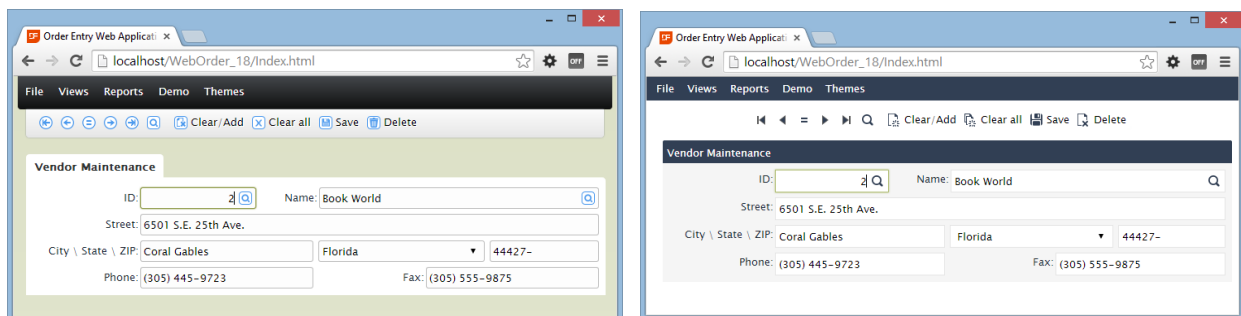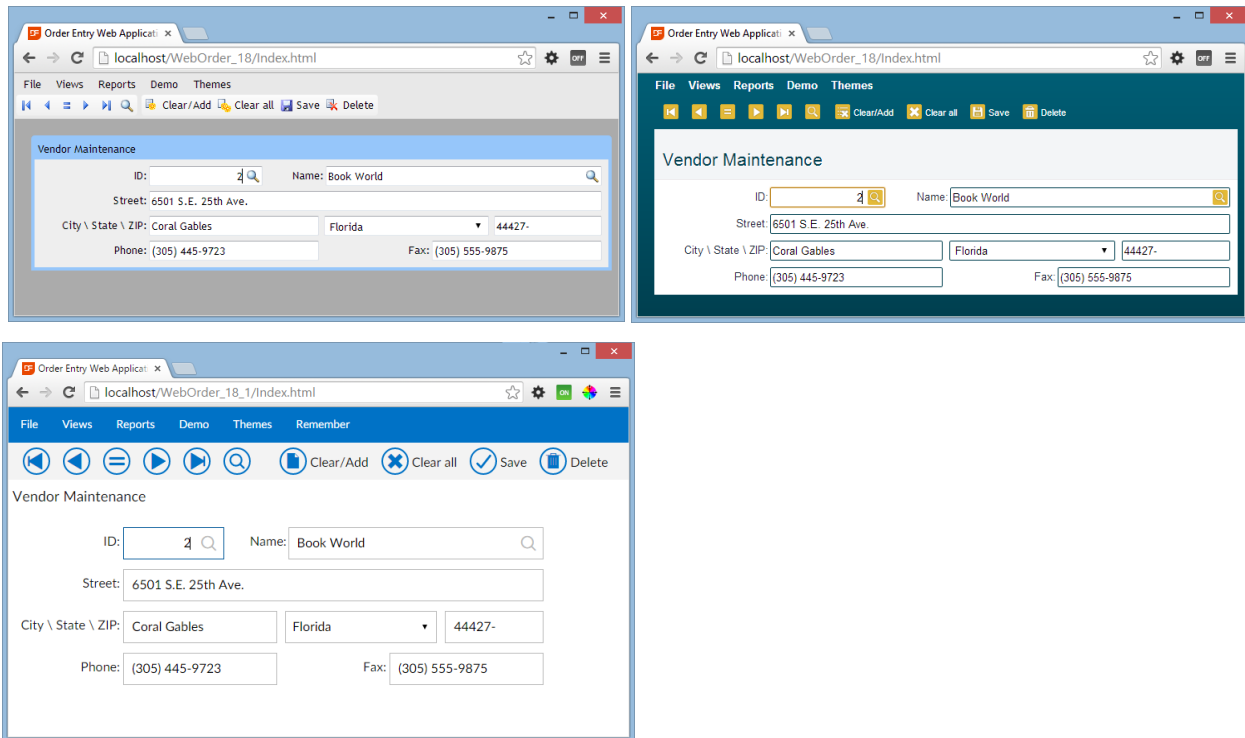
Database

DataFlex can work with any popular database management system. For instance, the combination of DataFlex Personal and Microsoft SQL Express is very powerful. DataFlex comes with the necessary database drivers, but for our introduction we will limit ourselves to the embedded DataFlex database. Use the DataFlex Studio to maintain databases. The Studio allows for the creation of tables, the definition of business rules and custom coding.

At a later date, you can easily convert the tables in the native database to any other supported database backend. You may use the available DataFlex tools to automatically perform the conversion of existing data as well as table structures.

Data Dictionary

When entering data, you want to have the most accurate and consistent information stored in your database. Data Dictionaries help with validating the entered data. Examples of such validations are that the name of a State should be uppercase, or a customer may not order more than his credit-limit's amount. The name for those validations is 'Business Rules'. It makes sense to store these rules in one central place in what we call 'Data Dictionaries'. All applications automatically apply the rules when they use data dictionaries. Programmed rule changes are made in one place only. It also means that if you were to migrate the application to another database platform, the same business rules are automatically applied no matter what database backend is used. Create and maintain Data dictionaries via the Data Dictionary Modeler in the Studio.
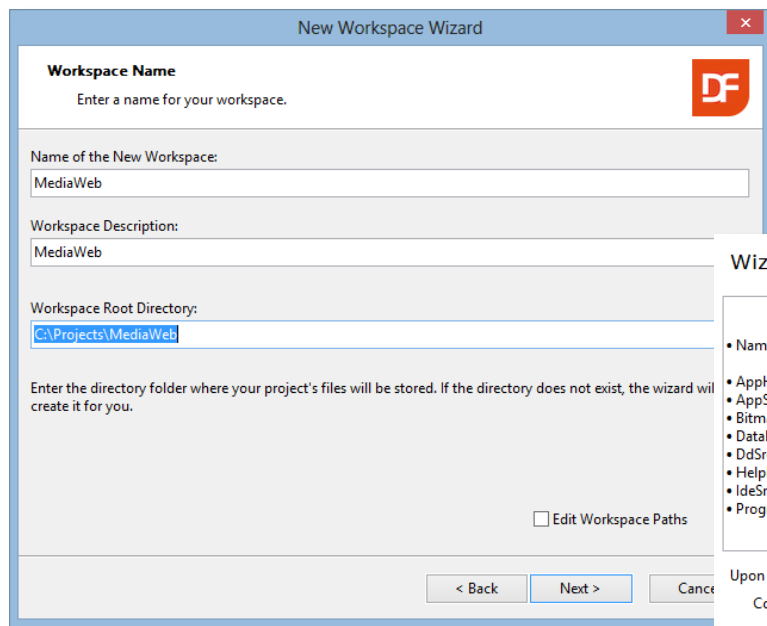
Our Example Scenario: Media

We will build a small database application that we will call Media. We will create a table Media in which we store all our CD's, DVD's, Books etc. Next, we will make a table named People to store the names of friends and relatives. To discover if you have the media yourself or lent to a friend linking of these two tables is required. In short, we will take you through the following steps:

- Create a project named Media.
  Create all components in a workspace while this project is active.

- Create a table named People.
  The table needs a unique key and columns to store address, phone-number, date of birth, etc. of a Person.
- Create the Person Data Entry Web View using the Data Entry Wizard to enter data into the People table.
  A web view is a web page to enter data.
- Compile the program and test our first results.
- Create a table named 'Media'. This will have a unique key and a few columns to store Author (/Artist/Writer), the media type and maybe the price and purchase date. In the table we also store the PeopleID, to be able to relate to the Person table.
- Manually, by using drag & drop, create a view to enter data into Media.

- After that, we will build in some more advanced features:
  - Make sure that the Media- and People ID's are automatically generated sequential numbers.
  - Put user-friendly pop-up lists (selection lists) in the application to easily and quickly find records in Media and People.
  - Enter the column 'Media.Type' in a consistent format. We will create a combo box for it and make sure the user always enters the types in a consistent manner.
  - Enable the user to store a picture with Media.
  - Create a picture browser
  - Create a module to discover which People own/borrowed what Media.
  - Create a module to search with wild cards.
  - Create a DataFlex Reports report and integrate the report in the project.

Getting Started!

Start the DataFlex Studio. If you use Windows 7, 8 or 10 please start the Studio via "Run as administrator". This is available via a right mouse click on the menu option. Once the first steps are taken a normal start is enough. We will start by creating a new workspace. From the File menu, choose New Workspace… Give the workspace a name - in our example, we

will name it MediaWeb and store it under C:\Projects\MediaWeb. The folder should allow web shares and Windows has strict rules about the location.

After clicking the Next button accept all defaults in the Database Type wizard page as we use the embedded database.

Prior to closing the wizard you will see a summary of the gathered information and what to do next.
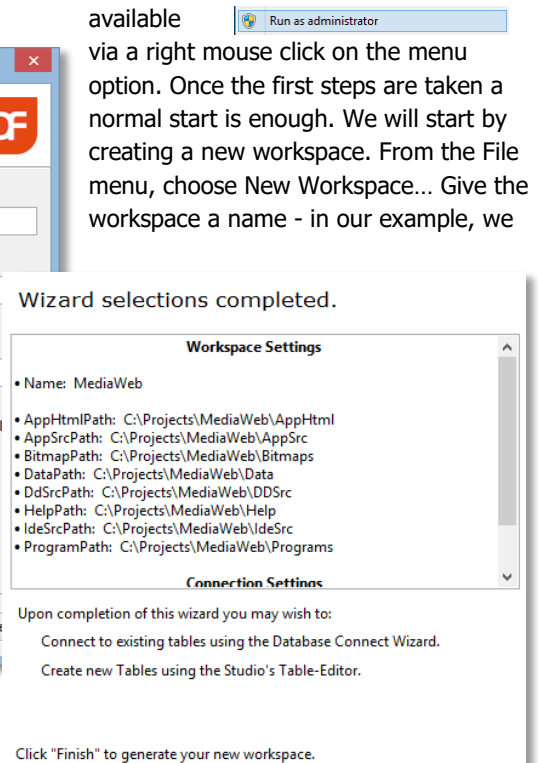
The wizard creates the folders for the Media workspace and returns to the Studio so that you take the next steps.

Note: Workspace information can be altered later via the DataFlex Studio and – if folder renaming is desired – the Windows Explorer.

The Dashboard

After the workspace has been created the DataFlex Studio will open the workspace dashboard. The dashboard is feature that guides you through the application development. The dashboard gathers information from the workspace and shows where you might want to pay attention to. A red colored marker indicates a required action, a yellow colored marker indicates that you need to pay attention to this group of items.

The dashboard as shown here says that the next step is either the table or project creation. We will first create the project.

Tip: *Keep the dashboard open and notice that it will automatically update during the whole process of application development.*

Tip: *At any time in the project development you can add TODO markers which are picked up by the dashboard, this means you can use the dashboard as a project management tool.*
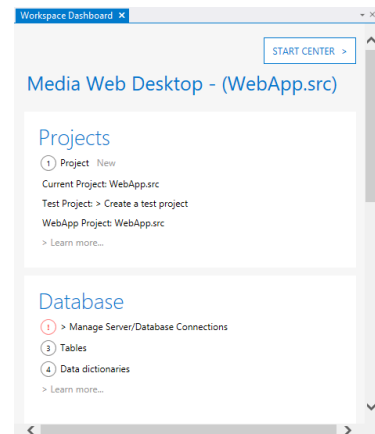
Create the Desktop Web Project

The next step is to create a project. For almost everything we create in the Studio we need an active project. There are several ways in the Studio from which you can create a project. For now, click the option in the dashboard. Alternatively, you can also create a new project via the File pull-down menu, option New, and Project.

This will open a dialog in which we select "Desktop Web Project", usually the third option in the set of icons. This choice displays a dialog where you then enter the name of the application and virtual directory. The default names values are identical to the workspace name. These names must be unique on the same machine. The virtual directory name, used for the URL, is an attribute from Microsoft IIS. Later you will see a URL http://localhost/MediaWebDesktop and IIS uses the MediaWebDesktop part of the name to find the web application.

It is not possible to change the directory name in this dialog. If you would like a different folder name you should have indicated this in the new workspace wizard. The default will be good enough to work.

A workspace can only have one web application and to make it easy the Studio uses the name WebApp.src which cannot be changed here.

Click OK now; a progress panel shows the files being copied. Finally, the Studio creates the WebApp.Src file on disk and makes the WebApp.src the current project. The project file contains two main objects one being a cWebApp containing a menu structure. We can already press the F5 key to run the application but we better first focus on the creation of a table in which we will store the data.

Note: *It is the creation of the web application that requires you to start the Studio as Administrator. From here on, now that the application is created, it is no longer required.*
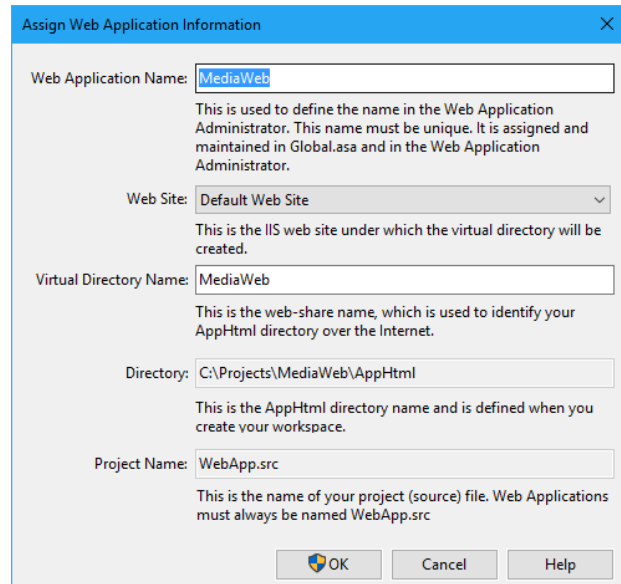
Create the People Table

The next step in the process is to create one or more tables. Use the Table Explorer to create new tables. As usual, there are several ways to come to the point to start the table creation. Make sure you have the Table Explorer window open. If Table Explorer is not opened – by default you will find this window on the left hand side of the screen, grouped together with Code Explorer – you can activate it via the View pull-down menu, Table Explorer option or the button positioned in the views toolbar.
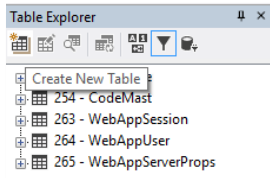
The Table Explorer panel consists of a tool-bar and a list (tree-view) which shows the tables already present in your workspace. A file called filelist.cfg contains the names of the tables. This file is automatically maintained for you.
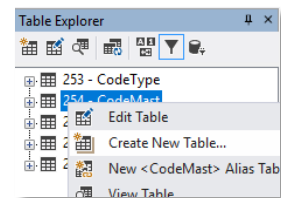
Use the buttons above the list to create, drop or open a table for modification. A floating menu, open this with a right mouse click in the Table Explorer window, offers the same functionality.

In the floating menu you will notice a couple of data dictionary options. We will discuss the use of the data dictionaries later.

So click the "Create New Table" button (first button) or choose the "Create New Table" option from the floating menu.
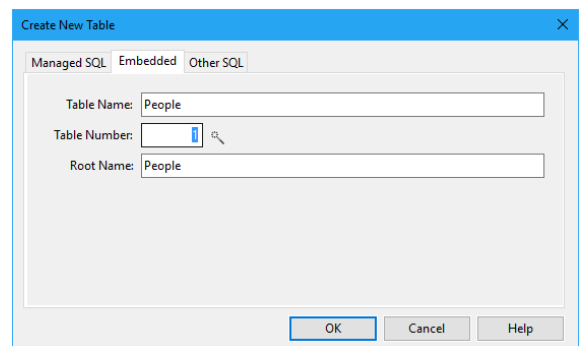
In the dialog you should enter the name of the table – People – in two of the input fields (labeled Table Name and Root Name). The other parts of the dialog are not important at this moment; they are for more advanced use.

Pressing the OK button will instruct the DataFlex Studio to open a Table Editor for the new table we are creating. The Table Editor panel contains areas with Columns, Indexes and Relationships information.

The column information is editable via a grid in which you can specify the names of the columns and their type, length and main index. Create the table to match the following screenshot.

| Name	Type	Size	Main Index
PeopleId	Numeric	6,0	
LastName	ASCII	40,0	
FirstName	ASCII	30,0	
Address	ASCII	40,0	
Zip	ASCII	8,0	
City	ASCII	40,0	
Phone	ASCII	25,0	
Comments	Text	1024,0	
		,0	

If you would like to, enter more columns; you might want to store the size of their shoes, their hobbies or an e-mail address. Feel free to do so. The quick introduction assumes you have created the shown columns of the given type and size. To identify a specific row in the People table create the column PeopleId as a key-field.

In order to look up records, we will create a couple of indexes. Let us suppose we want to allow to search by LastName, FirstName and Zip. We need to create an index for each one of them and each index need to be unique by itself.

The picture shows that the second index consists of two segments: LastName and PeopleID, the latter making the index unique. Also notice that the checkbox Ignore Case is checked. This means that when a user searches on "Johnson" the order in which it is found is not dependent on whether it is typed as "JOHNSON", "johnson" or "Johnson".

The tab has two toolbars. One - with the buttons "Add Index" and "Delete Index" – and this works on the list of indexes making it possible for you to add or remove a complete index while the other toolbar works on the grid with index segments. Change the order of index segments with the "promote" and "demote" tool-bar buttons if they are in the wrong order.

Save the table structure for the table People by pressing the Ctrl+s key-combination or click the save tool-bar button.
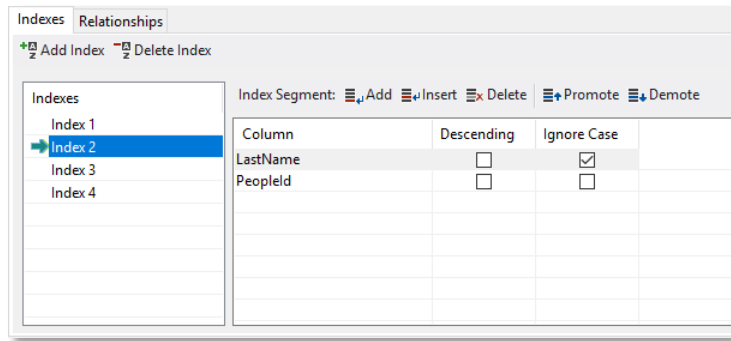
The Business Rules

When a column is marked as a key-field ('Protect value (Key)' attribute) the value cannot be changed after the record has been created. PeopleID is used to link the rows between the later to be created Media table and People table and for this reason we do not want to allow this value to be changed. To indicate that the PeopleId column is a key-field, we need to modify a setting in the data dictionary for the table People. While the data dictionary class is automatically created when we created the table, it is not opened for editing yet. To open the data dictionary, right click the table in the table editor and select "Open Data Dictionary" from the menu.
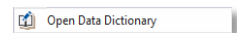
One new tab-page in the code editor part of the DataFlex Studio will open and the focus will be on the DD modeling tab.
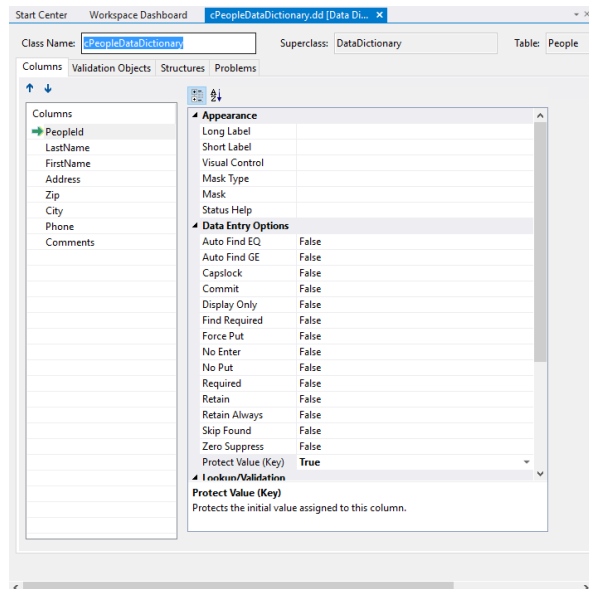
Click the PeopleId column in the list of columns and find the option "Protect Value (Key)" in the list of properties as shown to the right. Change the value of this setting from **False** to **True** and we've finished setting the key field.

While we are on this screen we can also add a couple more Business Rules:
- Make sure that the column LastName is always entered – select LastName from the columns list and set its Required attribute to True.
- Make sure that the Zip and City are always stored in uppercase – the same for Zip and City, changing the Capslock attribute to True.

We need to save the table and data dictionary to disk. Either save each one independently, or use the Save all option. If you select the Save all option you also save changed source code which might be a good idea anyway. You may still undo the change after saving as it has no influence on the undo stack. On the other hand, closing a file would affect the stack and you may not undo all the changes.

Creating the People Data Entry View

We can now create a data entry view and we will do so with the Web View Wizard. Again click on File, New and now Web Object and the panel shown to the right appears.

As you can see, there are a number of wizards and templates available. Choose the 'Web View Wizard' icon.

Enter the following information while processing the wizard:

- Enter oPeopleView for the object name.
- Enter PeopleView as filename.
- Enter People for the description.
- Choose 'Simple view.
- Choose the People

Form Web View' to create a simple data entry web

table.

As shown, place all the columns on the View. By adding two tab-pages, we can create a web view with the related information items grouped together. On top of that there will be more space for entering comments.
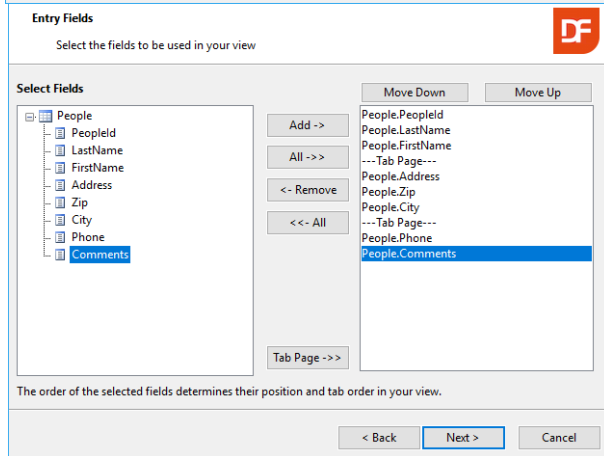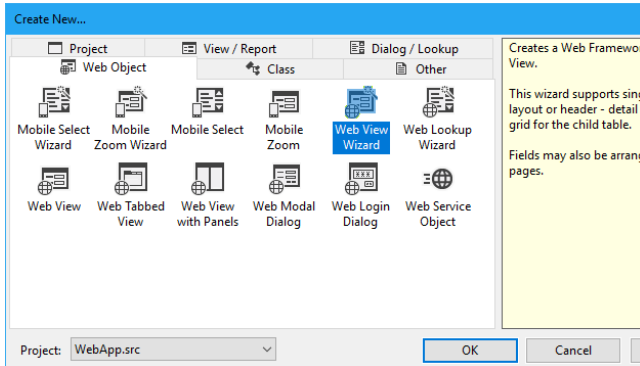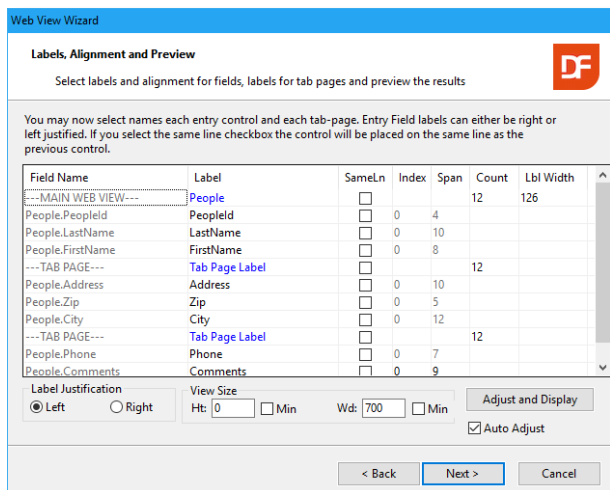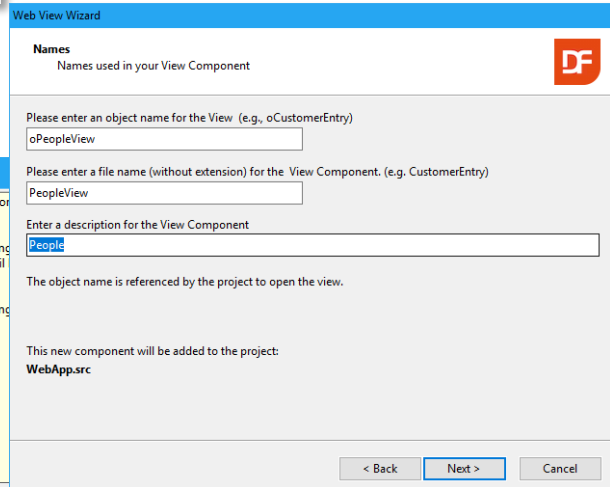

On the next wizard page you can indicate whether you want to see the labels aligned left (always placed on the left hand side of the controls) or right and change the text of each label. Change the labels for the tab-pages.

You can click the button labeled "Adjust and Display" to see what your web data entry view would look like. Web controls are laid out in a column based structure (see the chapter "Layout and Positioning"). The number of columns (automatic or self calculated) determine the width of an input control and the amount of controls that can be placed on a horizontal line. Later – after the wizard finished the web component – it is possible to change all the settings. We recommend using the default values at this moment.

Click Next and Finish the wizard which will bring you back to the Studio. In the Studio, the result of the wizard will be loaded automatically. You will see the source code.

![DataFlex](DF DataFlex)

Press [F7] to view the layout in the WebApp Designer. By changing object properties you can change the layout, change labels and more. The object properties are displayed in a panel that can be activated by pressing the [Ctrl+2] key-combination (or via the menu-item View, Properties). If you did not change the labels of the tab-pages, now is a good moment to change it. To do this:
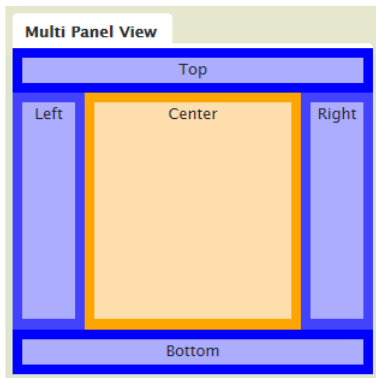
- Expand the object structure in the code explorer panel.
- Locate the oPage1 object.
- Switch to the properties panel.
- Locate the psCaption property.
- Change the text of the first tab-page to "Address"
- Do the same – different label value – for the second tab-page.
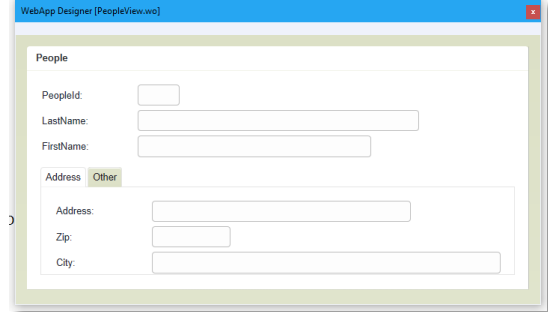
Layout and positioning

Before continuing developing the application let us take a look at how the DataFlex Web framework uses a column and panel layout system to divide the available space in the browser for positioning and sizing of the HTML objects. Let us take a look at the just created oPeopleView object in the code explorer window. Notice the view contains one panel (oWebMainPanel) divided into three input controls and a tab-container with two tab-pages. Each tab-page has a couple input controls.

Let us first focus on the panel. It is possible to divide each view – but also each panel into a maximum of five panels. There can be one top, one bottom, one left, one right and one center panel, controlled by the property named peRegion. In the oPeopleView there is only one panel and it is a center panel.

Each panel divides into columns, often set to 12, making positioning reasonable easy. Each HTML object can start in one of the columns (piColumnIndex) and can span a number of columns (piColumnSpan). If piColumnSpan is set to 0 is tells the system to take all the columns defined in the panel. The first column is column 0. If you want to combine two objects at the same "line" they must divide the available number of columns of the panel amongst each other. This does not need to be done evenly.

The creation order of the objects and their column index / span determines the position of the objects.
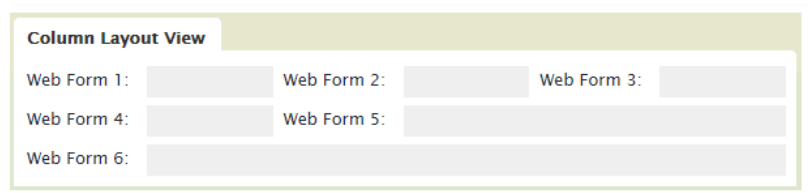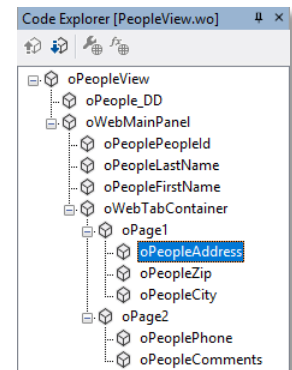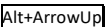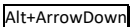
WebApp Designer Panel

After you have finished the wizard your View will be shown in the code editor. The Studio shows the created view in the WebApp Designer as well. Press the [F7] key if the designer is not present. Toggle the designer's visible state on and off via this key.

Select a 'control' via a mouse click in an input field or a container. The designer and the code explorer show the selected control. A control editor bar appears above the selected field or container.

If the selected control is a container the control editor shows the number of columns (piColumnCount). Use the `− +` buttons to increase or decrease this value.

If the selected control is an input field the control editor shows a different bar. Use the `<↔>` button to change the starting column of the control (piColumnIndex).
Use the `<>` button to change the number of columns used for the control (piColumnSpan).

Finally, the control can be dragged to a different spot in the layout. This changes the object creation – and tab – order. While dragging the designer shows an I-beam to indicate the insertion point. Alternatively, it is possible to change the object order via the code explorer. Use the `Alt+ArrowUp` and `Alt+ArrowDown` key combinations.

Note: It is not possible to change the column index by dragging the object.

Testing

You are now ready to test your DataFlex Web application. Compile the project to do so. Based on the generated source-code, the compiler will make an executable (webapp.exe). Start the application after a successful compilation. Select one of the following four ways to start testing:

1. Press `F8` to only compile.
2. Press `F5` to run. This launches the compiler if needed.
3. Choose the Project pull-down and select Compile.
4. Click on the little green triangle icon in the "debug" tool-bar.

If you selected option 2 or 4 and the compilation is finished, the application starts by opening or attaching to your preferred web browser.

The application starts with a login box in which you can enter "guest" for the login name and the password. We tell you more about the login soon.
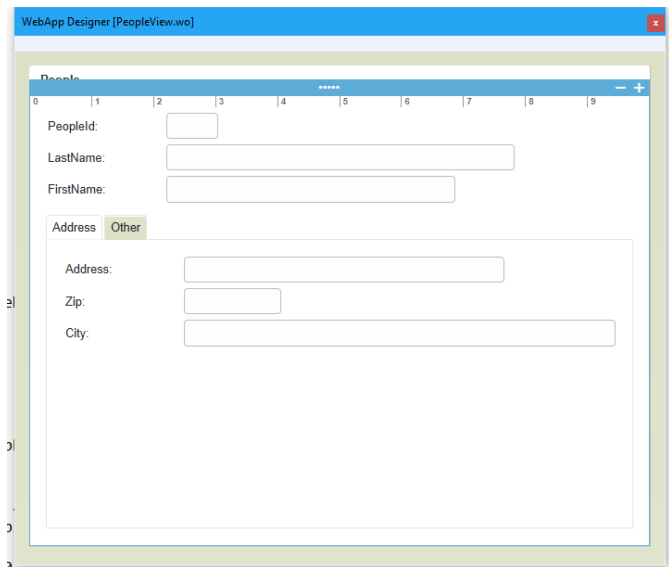
You can configure the DataFlex Studio to not opening your browser (Internet Explorer, FireFox, Chrome etc) each time the F5 key (or run button) is pressed. Open the application only once via the context menu in the workspace explorer and refresh the browser window when the application was changed and layout changes need to picked up by the browser.
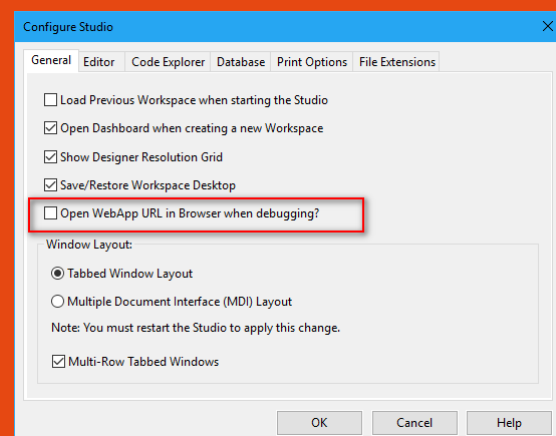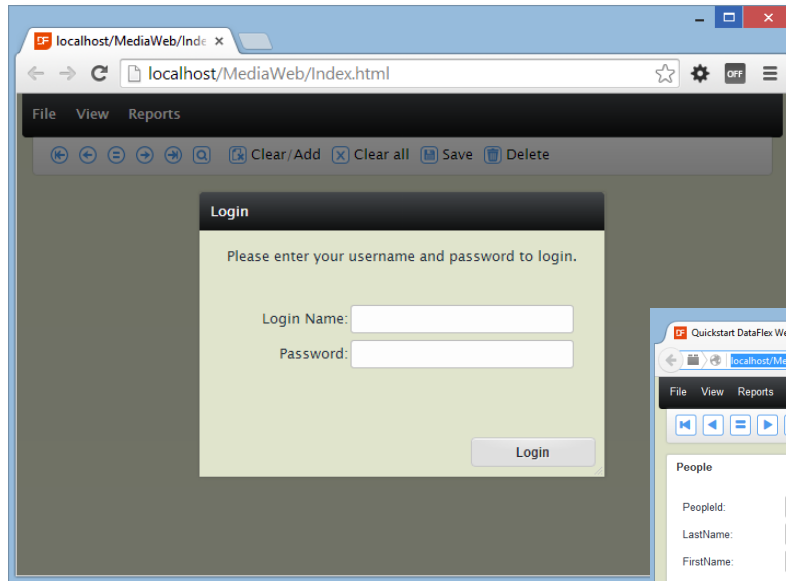
Now bring up the People view (from the view pull-down) and enter some records. Notice the following:

• Begin with the first record by using PeopleId '1'. The next is '2' and so on (more about this later).
• Save new People can be done by pressing the [F2] key, or clicking the save icon on the tool-bar.

• Clearing the screen can be done by [F5] (the current People row will <u>not</u> be deleted).
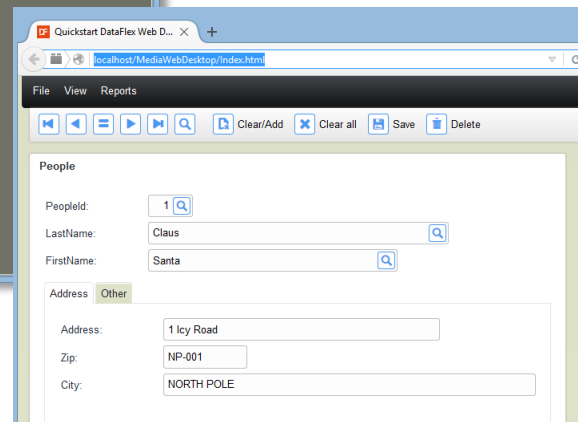• Once you have entered multiple rows, you can browse through them by using [F7] (Previous) and [F8] (Next). [F7] and [F8] work only if the cursor is placed in the columns PeopleID, LastName, FirstName or City. That is makes sense: These were the columns that we defined indexes on.
• Key in a partial name in LastName and hit [F9]. This will find (equal) based on the given (partial) string. Try this!

Close the browser application to return to the Studio.

Note that sometimes closing the debugger does not close the run mode of the Studio. You can stop the running mode by clicking the stop button.

Login system

As you have seen in the preview chapter the application opens by showing a login screen. Each DataFlex web application contains a session management system consisting of a user and a session table. The session table is a requirement but the login is not. You might want to allow everyone to use the application, or offer special features after login under a different user account.

To make testing easier you can turn off the login during the development phase of the web application. To do so make webapp.src the current tab-page in the DataFlex Studio and click on the oWebApp object in the code explorer. In the properties panel locate the peLoginMode property and change this from lmLoginRequired to lmLoginNone. For applications that offer additional functionality after logging in the lmLoginSupported option is available.
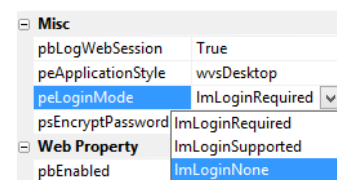
Media table

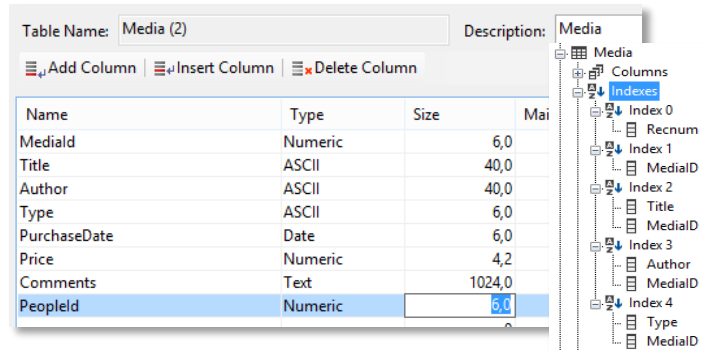The creation of the table for the Media is the next step in our process. So click again the New Table button in Table Explorer. Enter the value "Media" for the table and the root names. Then create the columns as shown in the picture. Note:

·   In the column Type we want to keep track of if it's a CD, DVD, BOOK etc.
·   The PurchaseDate is of the type Date.
·   The Price is numeric with the 4.2 format. This indicates that the price can have four digits before and two after the decimal point.
·   We will use the column PeopleID to connect Media with People. We will, therefore, ensure that it is of the same type (numeric) and same size (6 digits) as the column in the other table.
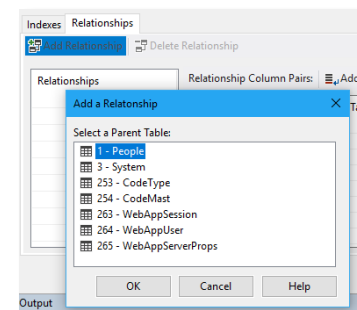
MediaID will be the key field. Besides that, create indexes on Title, Author and Type. To make them unique, we add MediaID as last segment of each index. We also choose to switch on the Case Insensitive option.

Tip: *Until now we have explained that an index is there to quickly and easily find a record. Indexes have another important function, which is fast sorting in reports. It is not difficult to create more indexes at a later time if you need this for certain reports.*

The Media table contains information about records of our media in the possession of a certain person. In technical

terms this means there is a relationship between the tables Media and People. Therefore, let us create the relationship between the two tables. Choose the tab-page named Relationships and choose the first tool-bar button (Add relationship). A dialog with tables pops up and you should select the table People from this list. Relationships are almost always defined from many to one, so in this case, from Media to People.

The selection of the parent table opens the option to specify from which child column(s) to which parent column(s) the relationship is made. The type and length of the related columns must match and the parent column (usually the key-field) must be uniquely indexed. The current table layout meets these criteria.
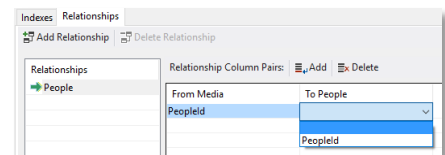
The Business Rules for the Media table

Finally, we will add some more Business Rules in the Data Dictionary. For this, the table needs to be saved first. The MediaID column will be a Key Field and the Title column is required. You should be able to do this with the guidelines given with the People data dictionary.
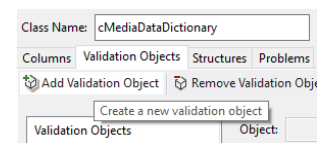
The values for Type should always be entered in uppercase (the Capslock attribute needs to be selected), but let's add something extra. We want the user to use consistent naming when entering Media Types. Enter 'CD' if it is a CD-Rom, enter 'BOOK' if it is a book. Not consistently entering such details makes report selections (such as 'Show me all books') quite difficult. Therefore, we will make a simple validation table on the column Type. You do this via the Validation Objects tab-page.

Click the "Add Validation Object" button select the 'Type'
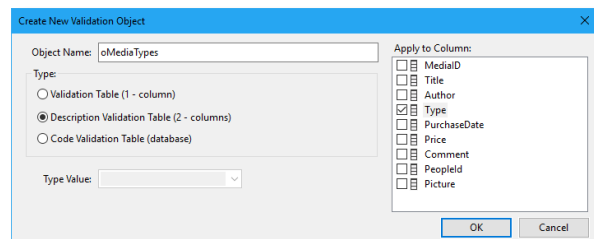
column under 'Apply to Column' and 'Description Validation Table' for the type. Enter 'oMediaTypes' for the object name. Object names at this level need to have a unique name.

After you clicked the OK button you can start entering values for the table. We suggest you enter the values as shown.

Feel free to add more optional Types.

Via the Allow_Blank_State property of the oMediaTypes validation table you can indicate whether the value may be left blank or not. If the value is not set to True the user must select a value from the list when creating or editing a record. Make your own choice.

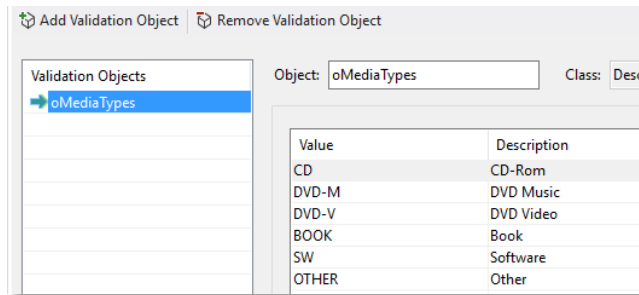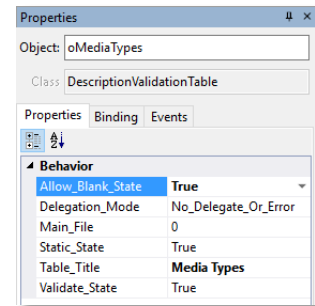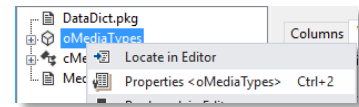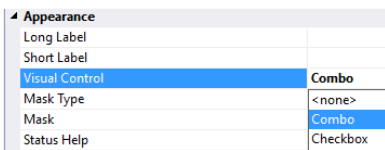To get a list of the media types in any web view to be constructed you have to change the visual control of the Type column in the Media data dictionary class.

Tip: *It is of no importance at this time, but open the tab-page called Structures where you can see that the People table obviously is added to the structure with Media. This is a hint that validations do not only apply to single tables; related tables are also validated.*

Creating the Media Data Entry View

The second view will be the Media Data Entry View. This time the wizard will not be used to create a data entry View. This means you will learn how to make a data entry view in a more manual way. From the menu under File, choose New, Web Object, but now click on: Web View.

After that, enter "oMediaView" for the object name and the file on disk will be named MediaView.wo.

Tip: *The dialog shows that the component will be added to the project WebApp.Src. It will also be placed in the menu of the application automatically.*
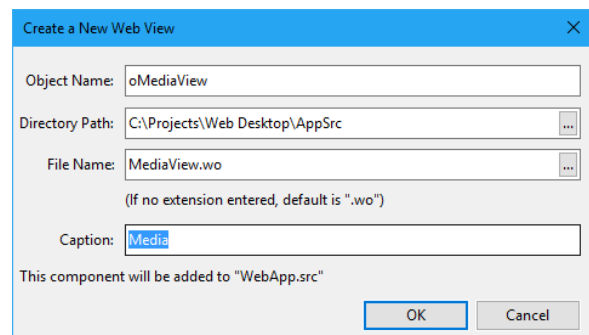
A new concept: Data Awareness

Before we continue, let's explain a new concept: **Data Awareness**. DataFlex is a tool to build database applications. After the first sample, we saw that it takes a View (interface) to enter data into the database. The several components in the interface are apparently coupled to the underlying columns in the tables. That is right, that's exactly what happened. But in fact there is an extra layer in between; the Data Dictionaries. DataFlex knows different types of components. An important difference is whether components are 'data aware', or not. If they are, you only have to assign Data Dictionary objects (DDO's) to it in order to have the desired data (tables) at your disposal. Data aware web controls make use of data binding usually via an Entry_Item statement.

To continue. You are now looking at a very basic cWebView in the Studio that contains a container, a dummy input control and some comment. The first thing we need to do is to decide which tables we want to maintain in this View. In the
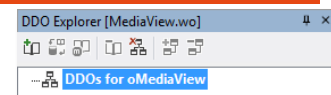
menu, under View, open DDO Explorer (or `Ctrl+4`). In DDO Explorer no tables are selected yet. Adding a DDO is done via clicking the "Add DDO" button. You can do the same from the floating menu (right click on "DDOs for..").

In the dialog that opens you have to select the right data dictionary for this new view. Since we want to edit (create, modify, delete) the table Media, select the cMediaDataDictionary class (highlighted in the picture).
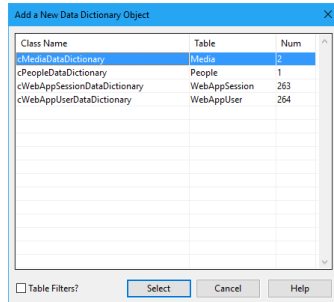
The Studio automatically makes the DDO for the Media table the main DDO and because Media has a relationship to a parent table (People), the DDO for that table will also be automatically added. When the choices are not correct, you can change these via the floating menu on each of the DDOs and apply the change. In our example this is now correct, so no action needed.

In the DDO Column Selector select all columns of Media (fill in all but the Recnum checkboxes) and drag them onto the view source code or into the WebApp Designer.

Insert them – as shown – inside the panel object just under "Set piColumnCount".
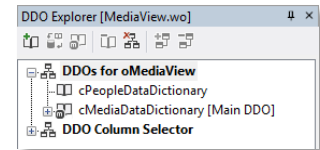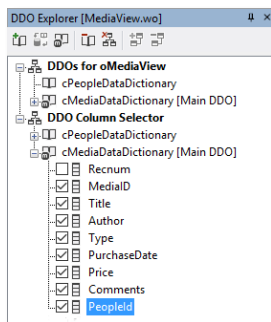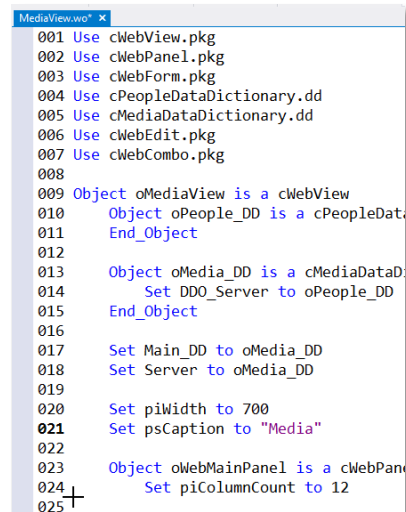
Similarly, drag & drop the column LastName from cPeopleDataDictionary onto the source just after the "PeopleId" cWebForm object just created during the first drag & drop.

Press the `F7` key and take a look at the preview window opened.

```
001  Use cWebView.pkg
002  Use cWebPanel.pkg
003  Use cWebForm.pkg
004  Use cPeopleDataDictionary.dd
005  Use cMediaDataDictionary.dd
006  Use cWebEdit.pkg
007  Use cWebCombo.pkg
008
009  Object oMediaView is a cWebView
010      Object oPeople_DD is a cPeopleData
011      End_Object
012
013      Object oMedia_DD is a cMediaDataD:
014          Set DDO_Server to oPeople_DD
015      End_Object
016
017      Set Main_DD to oMedia_DD
018      Set Server to oMedia_DD
019
020      Set piWidth to 700
021      Set psCaption to "Media"
022
023      Object oWebMainPanel is a cWebPan
024          Set piColumnCount to 12
025
```

Show LastName & FirstName Together

Wouldn't it be better if the People LastName input field could be positioned after the PeopleId input field, instead of underneath it? Yes, of course. You
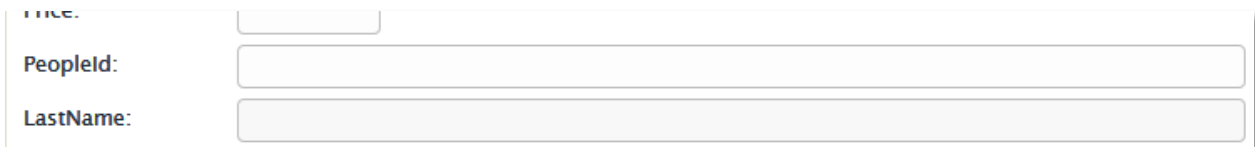
can do this if you understand the layout system a bit. Each container is divided into columns (also indicated in the data entry wizard). The default number of columns for a container is 12. An input like the PeopleId does not need 12 columns or the full width of the cWebView and that "line" can be shared with other controls, such as our "LastName" control.

There are three ways to change the number of columns used by the control:

- By selecting the object in the code explorer and change the piColumnSpan via the properties panel to 3
- By selecting the object via the WebApp Designer and use of the button.
- By locating the object in <> the code editor and directly change the piColumnSpan value. Tip: *click the object in the code explorer and select the "Locate in Editor" floating menu option.*

Use one of the above techniques for the oMedia_PeopleId object or try them all to see what you like the most.

The label "LastName" was removed by setting the pbShowLabel property to false.

To see what the application looks like after it has compiled and started you click the run button or press `F5`.

The new data entry view is labeled as specified in the create webview dialog (e.g. "Media") in the menu system. Enter a couple of media records. Let the Media ID start at '1' and choose the Person related to it by browsing through the Person records with `F7` and `F8`. Do this while the cursor sits in the PeopleId input field. Once you have created a few Media records, use `F7` and `F8` to browse through the data.

You might want to reduce the width of the MediaId, PurchaseDate and Price input controls (as shown above). To do this; change the piColumnSpan of these objects from zero (which means use all there is) to three or four. Use the code explorer to focus the object, use the properties panel to change the properties.

Date Selector

The input controls created by the wizard or the templates are "simple" controls. With a minimum of effort, you can get more power to the controls such as displaying an icon that a date can be selected from a date selector box. Of course, the date selector box opens when the user clicks the icon.

Locate in the source code the oMedia_PurchaseDate object and change the classname of the object to cWebDateForm. The code needs to be changed into the following:

```
Object oMedia_PurchaseDate is a cWebDateForm
    Entry_Item Media.PurchaseDate
```
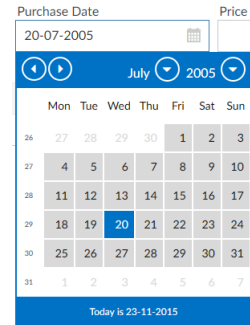
To complete the operation; jump to the top of the source code and insert the following line between the already present USE statements.

```
Use cWebDateForm.pkg
```

This addition makes the component autonomous and includes the necessary DataFlex class to make the control working.

More space for the Comments

It would make sense to give the comments input control more vertical space and therefore switch the location of the oMedia_PeopleId / oPeople_LastName controls in the object order with the oMedia_Comments object. Select one of the following three techniques:

- Select the oMedia_Comments object from "Object" to "End_Object", the press Ctrl+X to cut the code, move the insertion cursor in the code to the new location and press Ctrl+V
- Above is quite difficult for a Quickstart and the easier way is to locate the object oMedia_Comments in the code explorer and press the `Alt+DownArrow`) key combination twice. The first time the object will move between the objects oMedia_PeopleId and oPeople_LastName and the second time it will move the end. This object order change can also be done via the buttons in the tool-bar of the code explorer or via a menu choice in the floating menu of the code explorer panel
- The WebApp Designer offers the third way to move the object is. Select the object and drag it to the location behind the LastName object

Finally, to give more space to the Comments, locate the property pbFillHeight of the oMedia_Comments object and set this to true. This means that this object takes up all the vertical space left between the previous object and the bottom of its container. Each container can have multiple objects with the pbFillHeight set to true but it is better to limit this.

Displaying LastName and FirstName combined

Changing the behavior of the LastName WebForm control into showing both the LastName and FirstName values at the same time is a relatively easy job. For this select the oPeople_LastName object in the code explorer and then activate the properties panel (Ctrl+2). On the tab-page named "Events" double click the OnSetCalculatedValue event. The procedure appears in the source code. Now add "Move (People.LastName - ',' * People.FirstName) to sValue" in the procedure (it does not matter where as long as it is between "Procedure" and "End_Procedure" and that it is a full line of source code.

```
Object oPeople_LastName is a cWebForm
    Set piColumnSpan to 8
    Set piColumnIndex to 3
    Set pbEnabled to False
    Set psLabel to "LastName:"
    Set pbShowLabel to False

    Procedure OnSetCalculatedValue String ByRef sValue
        Forward Send OnSetCalculatedValue (&sValue)
        Move (People.LastName - ',' * People.FirstName) to sValue
    End_Procedure
End_Object
```

Set the pbEnabled property of the control to false because it makes no sense that a user can enter a value in a display control.

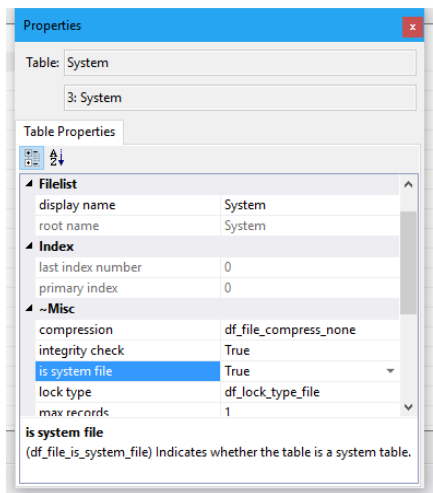*Make the Media view the default view*

If you feel the Media view is your main view which should be opened automatically after starting the application you can do this by adding "Set phoDefaultView to Self" to the code of the oMediaView object. Place it between the psCaption and the first cWebPanel object.

```
Set piWidth to 1000
Set piMinWidth to 1000
Set psCaption to "Media"

Set phoDefaultView to Self

Object oWebMainPanel is a cWebPanel
```

Automatically Generate Key Fields

For People as well as Media we defined unique, numeric keys. This number stored in those key fields is in fact not relevant to the user. In addition, it would be troublesome for users to remember what the last given number was when they try to create a new Media or Person record. This can easily be taken care of; it only takes two steps:

1. Create an extra table. In this table we will always store only one (1) record. This is called a system table. In this table we define two columns only: LastPeople and LastMedia. These columns are numeric, 6 digits.
2. The Data Dictionaries of Media and People will take care of generating these ID's automatically, using the system-file.

To make sure it actually becomes a system-file, select True for the "is system file" table attribute. Save the table structure before continuing.

We want the Data Dictionaries to automatically increment the ID each time a new record is saved. Open the data dictionaries for the tables People and Media in the Studio. Look for the attribute Auto Increment (grouped under Other) in the list of properties for the columns PeopleID (in People) and MediaID (in Media) and click the prompt button. From the list of tables, select System and from the columns the correct source data column – those are LastPeople for PeopleID and LastMedia for MediaID.
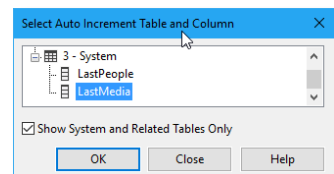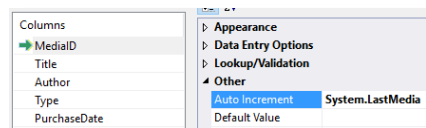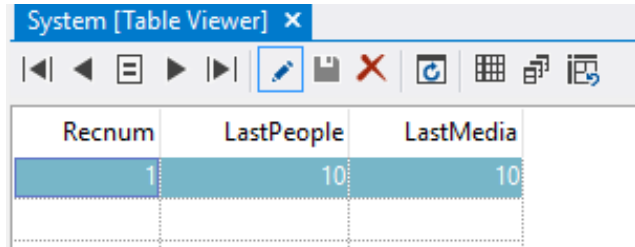
Note: The value can be taken from a system table or a parent table. That is why you see the checkbox labeled "Show System and Related Tables Only".

Ok. This should work, if not it is because you have already created some records, with ID's 1, 2, 3 etc. The first time we will use our automated increment function it will try save a record with ID of '1' and that won't work because that value already exists. ID's should always be unique – it is a key field! Our new application can't save any new records. We could delete our existing records, but there's another way to solve this. Right click the "System" table in the table explorer and choose "View Table". The contents of the table is opened in a tab-page in the design area of the Studio.

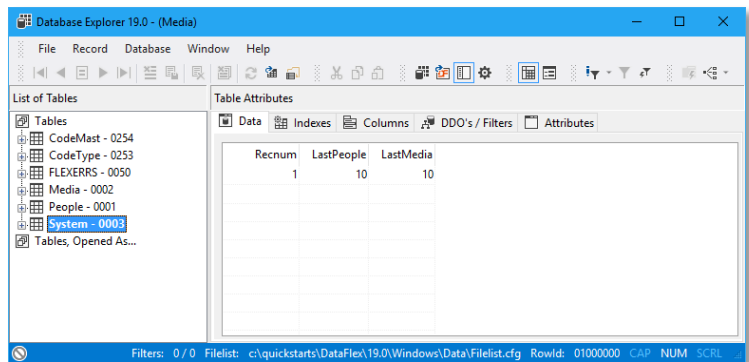Let's assume that the last ID you have entered was 10. Then enter the value 10 for both the LastPeople and LastMedia. Next time a record is created, the IDs will be automatically set to 11.

As an alternative you can do the same – and more – via another useful tool from the Studio Tools menu: Database Explorer.

Database Explorer (often called DBExplorer) provides a quick means to directly edit data in tables. It is a typical tool for developers, but be careful if you use it as it bypasses any safety or validations you have built into your applications.

The first thing we have to do is to make it possible to change data. By default, Database Explorer is configured in its most secure mode which is set to only allow us to read data. Therefore, click on the little icon at the very bottom-left. Change it from a red colored icon to a gray colored one. This is a one-time change; if you want to allow the read-write operation each time you open a table; open the configuration dialog and Flags, Table change the checkbox setting for "Open/Set Tables Readonly".

Data Dictionary changes

Before we compile and test our application, let's make a couple more improvements. Therefore open the Media and People data dictionaries if they are not already open.

Set the Auto Find EQ attribute In the Media data dictionary for the column MediaId to true. Do the same for PeopleId in the People data dictionary.

Locate the Status Help attribute and enter some status help text for some columns. You will see this appearing as tool-tip in the web pages. Use your own imagination.

Select the PhoneNumber in the People data dictionary and change the Capslock attribute to true. If you enter a phone number like 0800-CALLSANTA the characters will display uppercased.

Summary

We have made the following improvements:
· The ID's for People and Media are automatically serialized/incremented.
· Filling in an existing ID at Media, followed by pressing the Tab will automatically find the Media that goes with that ID, this is what Autofind EQ does. It makes sense to make the same change on ID for People.
· Entering data for Media Types now makes much more sense. Always uppercase, in a combo-box, which is the appropriate visual control for this type of data-entry.

- Setting the Appearance of Media.Type in the data dictionary to Combo-box, this column will always be displayed as a combo box by default.
- Help will be displayed for some items in the form of a tool-tip.

Selection Lists

We can make more improvements: If there are only a few records in the tables, it is not a problem to find the right data using F7 and F8, but when the tables grow, we must offer a better way for finding the right records, so we will add look-up lists, also known as Selection Lists.

From the File menu in Studio, choose New, Web Object and start-up the Web Lookup Wizard.
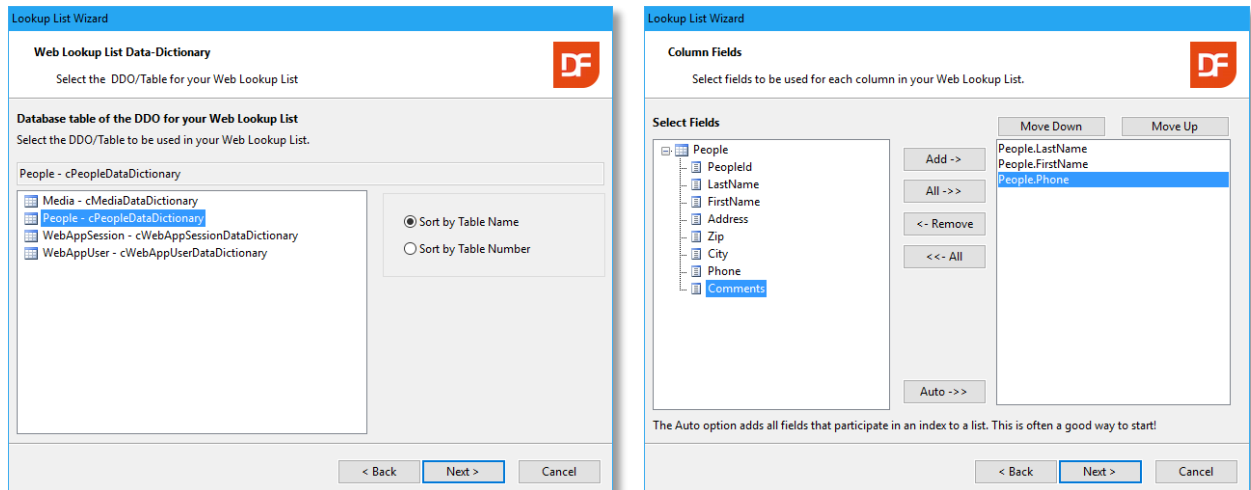
Let us first make a selection list for People, in which we place LastName, FirstName and Phone (number). Accept the defaults for the other wizard pages. Finish the wizard and compile/run the program again (F5).

You can see that buttons (Prompt Buttons) are automatically added to LastName and Firstname.
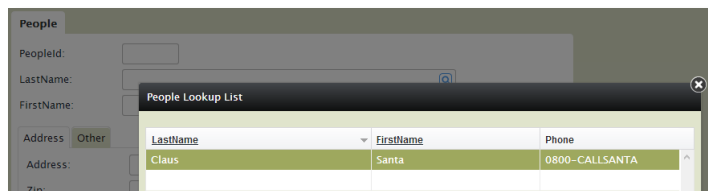
Clicking on the prompt button (or pressing F4) brings the just created selection list to the screen, offering the following standard functionality:

- Depending on the column in which the cursor is, the sort order of the list changes accordingly.

- Simply start typing the desired LastName and a pop-up screen appears. Pressing Enter will take you to the closest record. Because we use indexes, finding records in a set of just a few records will be as fast as when searching a record in a set of a few million!

Now it should be easy to create a selection list for the Media table. Maybe you want to try another way instead of using the wizard; by drag & drop. In that case, here are some tips: Choose File, New, Web Object: Web Modal Dialog. Enter "oMediaLookup" as the object name. The dialog contains two panel objects. In the top panel we will

create our prompt list object. Find the cWebPromptList entry in the class palette (Web Controls Group) and drag in source code inside the oMainPanel object. Remove the first automatic inserted cWebColumn object. Change the caption title to "Select Media" (psCaption). The generated name is oWebPrompList1, rename this to oWebPromptList.
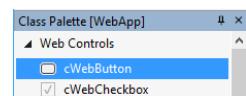
Use the DDO Selector to select the Media Data dictionary for the component. Drag the Title and Author from Media and LastName from People from the DDO Column Selector to the oWebPromptList object.

Now it becomes a bit trickier; you need to do some coding or copy, paste and change. The OK and CANCEL buttons from the template need to send their message to the oWebPromptList object. Therefore lookup the OnClick methods in the buttons and extend the "Send Ok" / "Send Cancel" lines with "of oWebPromptList". If you like to have a "Search" button you can copy the button from the People lookup component, adjust positioning of the three buttons and object reference.

```
Object oOkButton is a cWebButton
    Set psCaption to "OK"
    Set piColumnSpan to 1
    Set piColumnIndex to 3

    Procedure OnClick
        Send Ok of oWebPromptList
    End_Procedure

End_Object
```

In the bottom of the oMediaLookup object (cWebModalDialog class) you will find three methods and a lot of comments that need to be replaced with the following code:

```
Set pbServerOnShow to True
Procedure OnShow
    Send InitializePromptList of oWebPromptList
End_Procedure


Set pbServerOnSubmit to True
Procedure OnSubmit
    Send Ok of oWebPromptList
End_Procedure
```

One of the tasks automatically performed when using the Web Look up wizard is the connection of the lookup list with one or more columns from the table via the data dictionary. Since we did not use the wizard this time, we need to make these connections ourselves.

Open the Media data dictionary (if no longer opened), click the column(s) for which you want the oMediaLookup to appear and select the MediaLookup.wo file for the column property Web Lookup Object.

Once we have done this we can compile and run the application to see if we are happy with the results.

People lookup in Media View

If you look at the Media view it feels something is missing. There is a selection list for the Media but there is no selection list for the People record. To get that working, open the People data dictionary and select the PeopleId column. Now select the Web Lookup Object attribute and press the drop-down arrow. From the list select "oPeopleWebLookup" and save the data dictionary.

Compile and run the application and you can select a record from the People table via the selection list. One thing that is not working is the search button. Insert the following code in the oMedia_PeopleId object to get this working:

```
Procedure Prompt_Callback Integer hPrompt
    WebSet piInitialColumn of hPrompt to 0
End_Procedure
```

The selectionlist fires this event when it opens and you can use this to make run-time changes to settings that are normally design-time settings.

Show All Media Borrowed

Let us do something a bit more advanced. It is not so difficult to create a View where a user can browse through People and display in a grid all the Media that each person has borrowed. To browse through People and show only the appropriate Media, we need to create a view with a constraint, a filter.

Here is how you can do this:
- First, create a new index in the Media table containing the columns PeopleId, Title and MediaId. The new index is index number 5.
  - o Stop the web application via the web application administrator to edit this table definition.
  - o Save the table information
- Create a new web view using the Web View Wizard.

- Name the object oWhoHasWhatView.
- Use the "Header Detail Web View" option.
- Select the Media data dictionary in the child DDO page.
- Select the People data dictionary (there is only one class) in the header DDO page.
- Select LastName and FirstName as the header entry fields.
- Select Title, Author and Type for the detail entry fields.
- On the labels and alignment wizard page:
  - Deselect "Auto Adjust".
  - Change the "Span" value for LastName from 10 to 5 and for FirstName from 8 to 5.
  - Tick Check the "SameLn" checkbox for the FirstName.
  - Change "Index" to 5 for the FirstName.
- Finish the wizard and the source code loads.
- Find the oDetailGrid object in the source code and change the class from cWebGrid to cWebList. This will make the grid read-only.
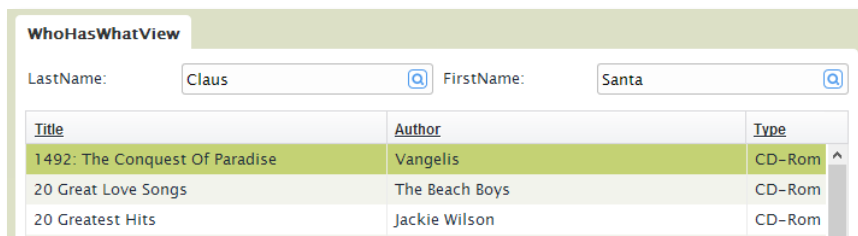
Pictures

The very last enhancement we make to the application is to add pictures for the Media we catalog. Let us say that we want to store a picture with each Media record. First, we need to change the table:

- Open the Media table if not open, and add a column named Picture.
- Specify the type ASCII, and a length of 255. The actual picture will not be stored here. Each record will hold a reference to the picture file name.
- Open the Media view again in Studio.
- Insert a cWebPanel (drag from the Class palette, under Web Containers) as sibling of the oWebMainPanel object. Name this object oWebPicturePanel (via properties panel or directly in the source code).
- If you currently have the WebApp Previewer panel opened, you will see an error. It tells you that you can only have one "main" panel. To correct this change the peRegion property of the object to prRight. This will also divide the web view into two containers side-by-side.
- Set the width of this new container to 250. Normally you don't set the width of containers but in this case we want to make space for the picture.
- Change the width of the cWebView from 700 to 1000.
- Set piMinWidth to 1000.
- Drag the new picture field from the DDO Column Explorer to the oWebPicturePanel object.
- Change the name to oMedia_Picture_Hidden. Set the pbRender property of one of this object to false to really hide the object. This hidden object is needed to be able to store the filename of the picture in the record.
- Drag a cWebImage object from the class palette to the cWebPicturePanel object. Name this object oMedia_Picture. Hide the label of this object (pbShowLabel to false).

The cWebImage class is not data-aware and we need to do some coding to display and save a picture. Before we do that we need to decide where the pictures stored. The easiest way is to store them in the web-shared folder (or one of the sub-folders) but this means that the files can be accessed outside the application. That is not a big issue for the pictures of this Media application but if the data is more sensitive (employee photos, documents, reports etc.) it is important to store them at an alternative location. You don't want someone to just get the files via a browser. The cWebImage class supports displaying an image from a secure and a shared location.

Displaying the image needs to be done based on an event. When the user browses to a different Media record the application sends two messages that could be used to respond to. We could respond on OnPostFind in the DDO or Refresh in the cWebImage object. The advantage of the Refresh method is that the code is directly related to the image control.

```
Procedure Refresh Integer eMode
    Boolean bSynching
    String sPath

    Forward Send Refresh eMode

    Get AppSynching to bSynching
    If (not (bSynching)) Begin
        Move (ExtractFilePath (Media.Picture)) to sPath
        If (sPath <> "") Begin
            Send UpdateLocalImage Media.Picture
        End
        Else Begin
            WebSet psUrl to ("Images/" - Media.Picture)
        End
    End
End_Procedure
```

The above code checks if the stored image file name includes a path or not. If a path is stored the control creates a secure URL to avoid the file can be accessed outside the application environment and if no path is present the image is supposed to be in a sub-folder of the web-share folder named Images.

The AppSynching check avoid that the code is executed when the application synchronizes itself when the browser connects to the application server.

Selecting an image from disk is not as easy as the rest of the application development we did so far. Should an image be selected from available images or should a new image upload be supported. Selecting from existing files requires that we need to determine to what folders on the server we would like the application user get access. To make an image selector – it is good to practice this – you have to do the following:

- Create a new modal dialog (File, New, Web Object, Web Modal Dialog). Name the object oPictureSelector. The dialog is the same kind of dialog you had to create for a selection list.
- In the dialog you will find two cWebPanel objects; create a third cWebPanel object and set its peRegion to prRight. Set the width of this panel to 250 (same as in the oMediaView).
- Drop a cWebImage object in this panel and set its height (piHeight) to 250.
- Drop a cWebList object in the main panel and name it oFilesList. Rename the column object to oFileNameColumn and change the caption to "FileName".
- Change the height of the list by setting the pbFillHeight to true. It will now take all the height of the container (cWebPanel).
- Make the cWebList not data aware by setting the property pbDataAware to false.
- The data needs to be passed to the list via an event called OnManualLoadData; add this event to the list via the events tab-page in the object properties.
- The first parameter of the OnManualLoadData is the most important one; it is an array that needs to be filled with files from the disk. To read the files from a folder on disk you can make use of the Direct_Input command. Use the code below to fill the list.

```
Procedure ScanFolder tWebRow[] ByRef aFiles String sFolder Integer iChannel
    Integer iRow
    String sFileName
```

```
    Move (SizeOfArray (aFiles)) to iRow

    Direct_Input channel iChannel ("DIR:" - sFolder - "\*.*")
    While (not (SeqEof))
        Readln channel iChannel sFileName
        If (not (SeqEof) and (Left (sFileName, 1) <> '[')) Begin
            Get EncryptKey Of ghoWebResourceManager (sFolder - '\' - Trim (sFileName)) ;
                to aFiles[iRow].aCells[0].sValue
            Move (Trim (sFileName)) to aFiles[iRow].aCells[1].sValue
            Increment iRow
        End
    Loop
    Close_Input channel iChannel
End_Procedure

Procedure OnManualLoadData tWebRow[] ByRef aFiles String ByRef sCurrentRowID
    String sFolder
    Integer iChannel
    Handle hoWorkspace

    Move (Seq_New_Channel ()) to iChannel
    If (iChannel >= 0) Begin
        Get phoWorkspace of ghoApplication to hoWorkspace
        Get psHome of hoWorkspace to sFolder
        If (Right (sFolder, 1) <> "\") Begin
            Move (sFolder - "\") to sFolder
        End
        Move (sFolder - "Images") to sFolder
        Send ScanFolder (&aFiles) sFolder iChannel

        // Now images in apphtml path
        Get psAppHtmlPath of hoWorkspace to sFolder
        If (Right (sFolder, 1) <> "\") Begin
            Move (sFolder - "\") to sFolder
        End
        Move (sFolder - "Images") to sFolder
        Send ScanFolder (&aFiles) sFolder iChannel

        Send Seq_Release_Channel iChannel
    End
End_Procedure
```

- If the user navigates thru the list of files it would be nice to show the image in the cWebImage object you have already created. To do this add:

```
Procedure OnChangeCurrentRow String sFromRowID String sToRowID
    String sFileName

    Forward Send OnChangeCurrentRow sFromRowID sToRowID

    WebSet psCurrentRowID to sToRowID
    Get DecryptKey of ghoWebResourceManager sToRowID to sFileName
    Send UpdateLocalImage of oWebImage sFileName
End_Procedure
```

- We are almost there...
- Find the OnSubmit event and add the oFileLists object as destination object for the Ok message. Add the same object reference to the Ok message send by the Ok button.
- Uncomment (Ctrl+K) the following line and change the code to:

```
Set pbServerOnShow to True

Procedure OnShow
    Send GridRefresh of oFilesList
End_Procedure
```

- One final addition; the dialog needs to have a function to return the name of the selected picture. For that add:

```
Function SelectedImage Returns String
    String sCurrentRowID sFileName

    WebGet psCurrentRowID of oFilesList to sCurrentRowID
    Get DecryptKey of ghoWebResourceManager sCurrentRowID to sFileName

    Function_Return sFileName
End_Function
```

Now we call (open) this dialog from the cWebImage object in the oMediaView object. Open the dialog in the OnClick event of the cWebImage object. Add the following code;

```
Set pbServerOnClick to True

Procedure OnClick
    Send Popup of oPictureSelector Self
End_Procedure
```

To use the selected picture add the following code, again to the cWebImage object in the oMediaView object;

```
Procedure OnCloseModalDialog Handle hoModalDialog
    String sFileName

    Get SelectedImage of hoModalDialog to sFileName
    Send UpdateLocalImage sFileName
    Set Field_Changed_Value of oMedia_DD Field Media.Picture to sFileName
End_Procedure
```

Finally, go to WebApp.Src, find the line "Use PictureSelector.wo" and move the line before the "Use MediaView.wo" line. Alternatively, you can add the "Use PictureSelector.wo" to the top of the code inside MediaView.wo.

Wildcard Search View

Wouldn't it be nice to have a web view where you can enter a text value that is used to filter the Media? Of course, you would like to have this! Here is how to do this.

- Create a new web view, use this time a Web Tabbed View. Suggested Name the object oMediaListView.
- Drop a cWebForm, a cWebButton, a cWebCheckbox and a cWebList object inside the first tab-page. Change the name of the cWebList object to oMediaList.
- Add a DDO for the Media table.
- Drag some data columns (Title, Author, Price…) from the DDO Column Selector (in the DDO Explorer) to the cWebList object.

- Align the button and the checkbox with the form on the same "line" giving the form more columns than the other two controls. For example; Set the piColumnSpan of the form to 8. Divide the rest of the default number of columns (= 12) between the checkbox and the button.
- Label the form "Filter on:", the checkbox "Case Sensitive" and the button "Search!".
- Name the form object oMediaSearch.
- Label the first tab-page "List" and the second tab-page "Details".
- Open the oMediaView – if not still opened – and copy the contents of the oWebMainPanel object into the second tab-page. This makes it possible to click on a row in the list, switch to the details tab-page and see/edit the Media details.

To get the list fills automatically when the view opens add the following code to the cWebView object of the oMediaListView object:

```
Set pbServerOnShow to true
Procedure OnShow
    Send FindFromTop of oMediaList
End_Procedure
```

The cWebView object sends this event when it opens.

To filter the data in the list we need to make use of the OnConstrain event in a DDO, in the oMedia_DD object to be precise. In this event we need to code the condition for the filter. We will make use of a "constrain as" technique. The use of "constrain as" should be avoided as the filter cannot be optimized but in this situation it is OK as long as the number of rows in the table is not too large.

Add the following code to the oMedia_DD object:

```
Procedure OnConstrain
    Constrain Media as (IsValidMediaRow (Self))
End_Procedure
```

Now write the following IsValidMediaRow method in the oMediaSearch object:

```
Function IsValidMediaRow Returns Boolean
    String sFilterValue sData
    Boolean bCaseSensitive bOk

    WebGet psFilterValue to sFilterValue
    WebGet pbCaseSensitive to bCaseSensitive

    Move (Trim (sFilterValue)) to sFilterValue
    If (sFilterValue <> "") Begin
        Move (Media.Title * Media.Author * Media.Type * String (Media.Price) * ;
            String (Media.PurchaseDate) * Media.Comments * Media.Picture) to sData

        If (not (bCaseSensitive)) Begin
            Move (Lowercase (sData) contains sFilterValue) to bOk
        End
        Else Begin
            Move (sData contains sFilterValue) to bOk
        End
    End
    Else Begin
        Move True to bOk
    End

    Function_Return bOk
```

```
End_Function
```

This method concatenates all columns we want to search and looks if the filter string is present in that value. You can increase or decrease the number of columns to compare with. The filter makes use of two self-defined properties (psFilterValue and pbCaseSensitive). Create them in the oMediaSearch object by adding:

```
{ WebProperty = True }
Property String psFilterValue
{ WebProperty = True }
Property Boolean pbCaseSensitive
```

The properties will get their value by clicking the search button. Change the contents of the button's OnClick event.

```
Send SetupFilters
Send FindFromTop of oMediaList
```

Add a self-defined method named SetupFilters to the oMediaSearch object. The code for this method is:

```
Procedure SetupFilters
    String sFilterValue
    Boolean bCaseSensitive

    WebGet psValue of oSearchForm to sFilterValue
    Get GetChecked of oCaseSensitiveCheckbox to bCaseSensitive

    If (not (bCaseSensitive)) Begin
        Move (Lowercase (sFilterValue)) to sFilterValue
    End

    WebSet psFilterValue to sFilterValue
    WebSet pbCaseSensitive to bCaseSensitive

    Send Rebuild_Constraints of oMedia_DD
End_Procedure
```
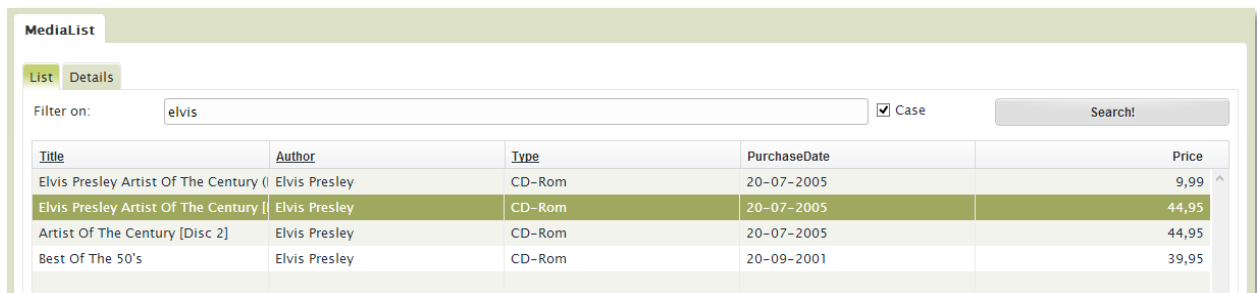
Compile and test your new search view. If you like you can add a checkbox to make case insensitive filtering possible (as shown in the screenshot).

Reports and Lists

One of the most powerful ways to create reports and integrate them in DataFlex is by using DataFlex Reports and the DataFlex Reports Integration Library. This wizard automatically integrates a report in your Web application. The end-user will be able to start the report from the menu and still be able to influence the sort order, output device, and even selection criteria. It's simple: First create a report with DataFlex Reports, and then start the wizard – the rest is self-explanatory.

Note: If you do not have a license for DataFlex Reports, please contact the Data Access sales representative in your region to receive an evaluation license, or to purchase a license.

Start DataFlex Reports and select File, New. Then choose Standard Report. You can also
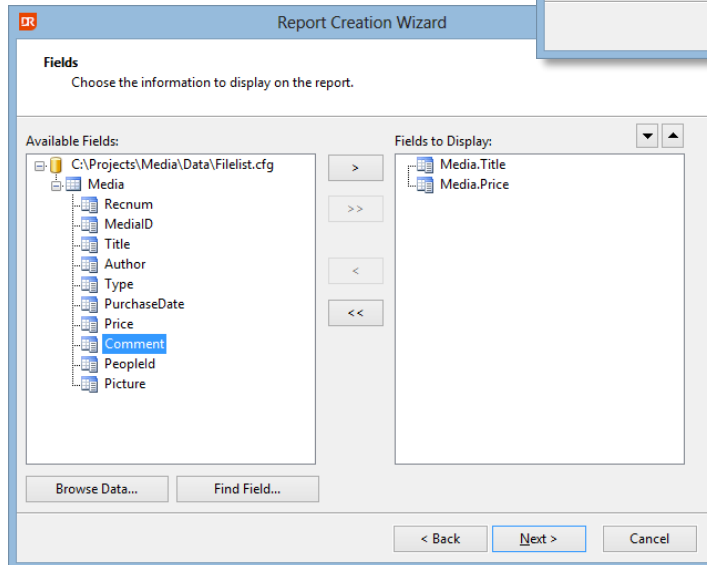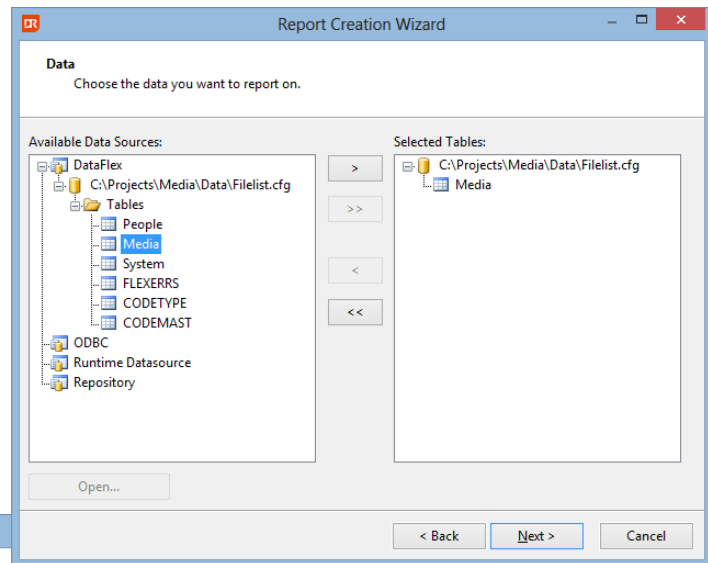
press the Ctrl+N key combination. In the wizard select DataFlex as your data source and point to the SWS file of your workspace (in the root folder). This will load the paths of the workspace and shows the contents of a file called the filelist. Select the Media table from the list of tables.

After clicking the 'Next' button you have to select which columns from the Media table should appear in the body section of the report. The body section of a report is repeated for each record that matches selection criteria. In the 'Fields' page select the columns Title and Price.

The next wizard page let you select the column(s) to group data on. Here we select the column Author. This means that printing of titles per author is possible.

Skip the summary, data filters and repository pages for this report.

After finishing you can preview the report in the designer and start making the first layout modifications.

Take a look at the screenshot and see that we used colors to 'beautify' the report. We also added a function to combine the Author name with the number of Media records we have for this author. The prices are summarized and finally the page footer contains a 'Page N of M' text.
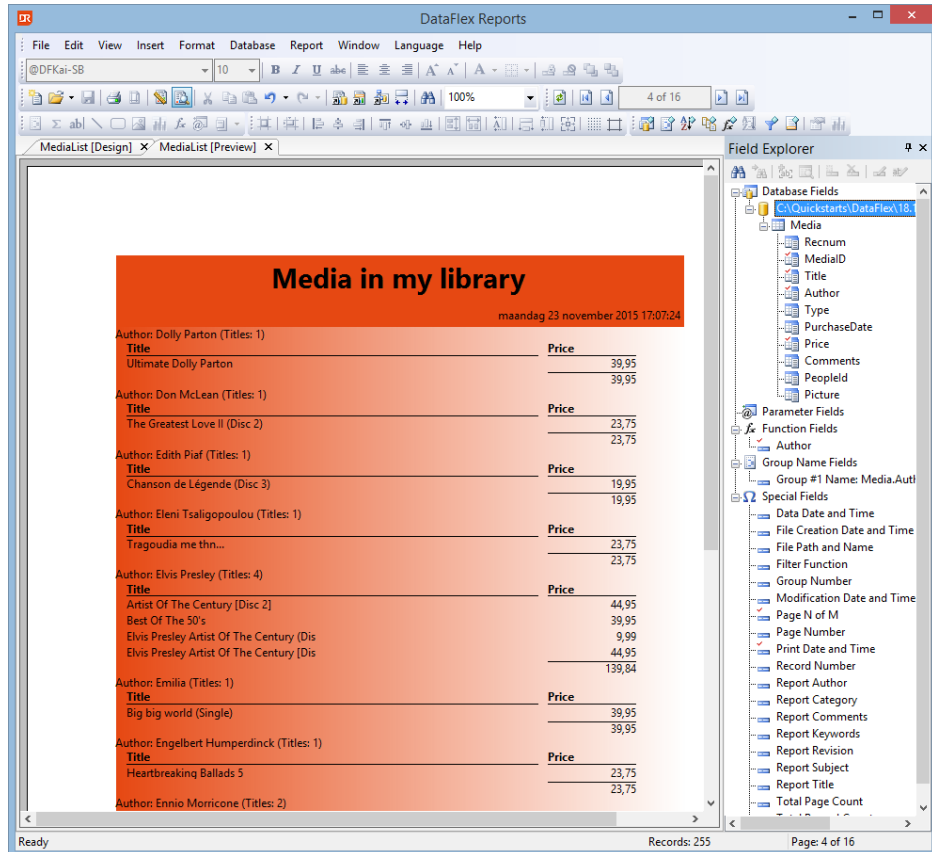
Not shown but interesting; we can sort the details of each group on Title or any other column.

DataFlex Reports developer edition optionally (turned on) installs an integration library. Attach to the library by selecting Tools, Maintain Libraries. In the dialog that pops up you will see the already connected libraries.

Press the "Add Library" button to select the SWS file of the DataFlex Reports integration library. The library is most likely installed in a folder libraries inside the DataFlex environment. If not look for \Libraries in the root of the installation disk (e.g. c:\libraries). The location is chosen during installation of DataFlex Reports developer edition. After clicking OK a wizard starts that guides you through the process of attaching. You should accept all the defaults. The wizard copies files and makes modifications to the main (often the index.html) file. Check if you can still compile and run your web application. It should still work!
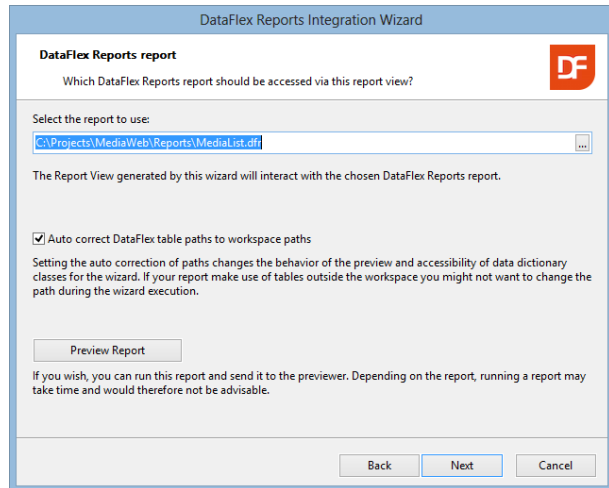
To integrate the report we start the integration wizard. Select File, New, Web Object and pick the DataFlex Reports Integration Wizard. Note that there is also a wizard for Windows programs but you need to one in the Web Object page.

In the wizard you can choose between connecting to an existing report or create a new RDS based report. Choose connect to an existing report. Then on the next page select the report that you want to integrate. You can preview the report but keep in mind that a preview might take some time when you have more data than present in this Quick introduction workspace.
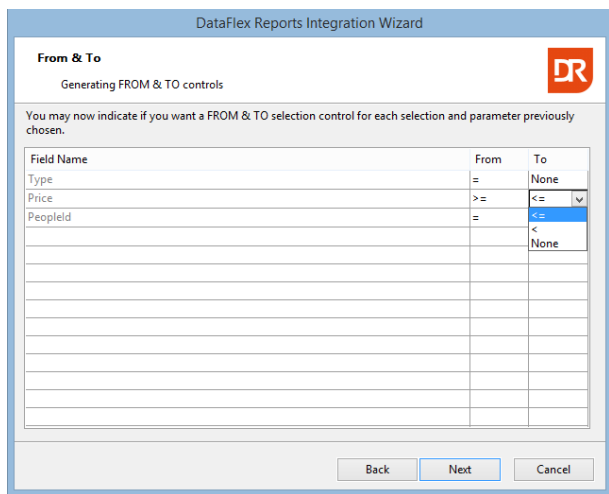
![DataFlex logo]

After clicking the 'Next' button you can select / confirm the application style. Use the 'Desktop' style option.

In the next wizard page you can choose to create user defined selection criteria controls. Based on the user input report data will be filtered. Select the columns Type, Price and PeopleId. Using selection criteria can be very important as you don't want to build a 500+ pages report. It takes too long for the average impatient web user and it also makes no sense to view so many pages over the web. You even might want to check if the user made selections or not.

Because we selected columns for selection criteria controls you can specify the values of the labels for the selection controls and whether the labels should be visible, aligned left, right or centered and positioned left or on top of the control on the next wizard page. Either accept the defaults or make a label text modification to see how this works.

Also based on selection criteria; if they are present you can specify the filter operator and whether you want a 'from-to' selection or selection on just one value.

Change the operator for the Price column to greater than or equals for the 'From' column and less than or equals in the 'To' column.
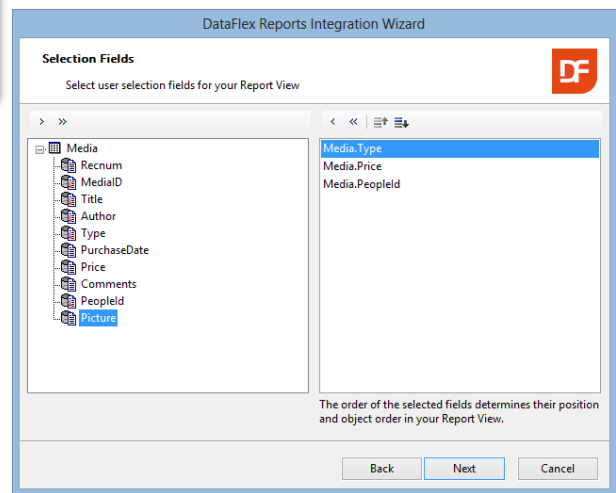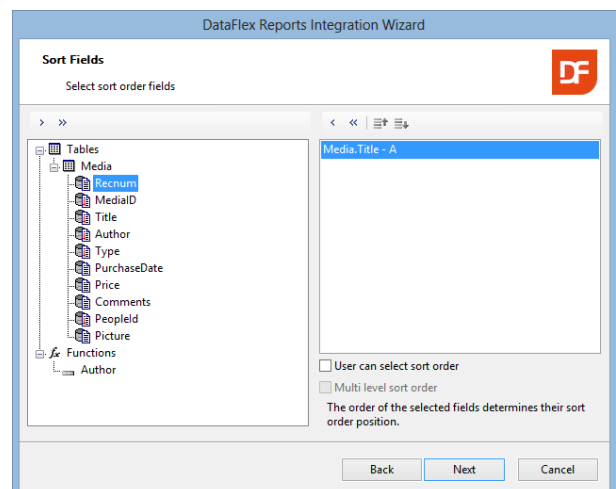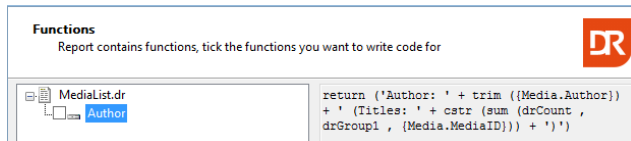
The next page shows the sort order defined in the report (if present). In the MediaList report this is the Title column. Add a couple of more sort fields like PurchaseDate and Price. Turn on the option that the user can change the sort order; if not turned on the data will be first sorted on Title, then on PurchaseDate and then on Price. With user selection the user can determine what sorting should be used. Optionally you can choose to generate a multi-level sort order control, for this report integration you should skip that.

If the report contains formulas you will see a wizard page showing the formulas and you can select whether

you want the wizard to generate code to change the content of a formula. The MediaList report contains one formula but it is not a candidate to be changed at runtime.

The next important wizard page is the output selection. Here you can choose from the supported export formats (PDF, Image, HTML, Excel, Word and CSV). Base on the first choice the destination drop-down will change and offer more or less destination options.

For (Desktop style) web applications the default will be PDF displayed in the page. You might want to use one of the other options such as HTML via an HTML previewer control.
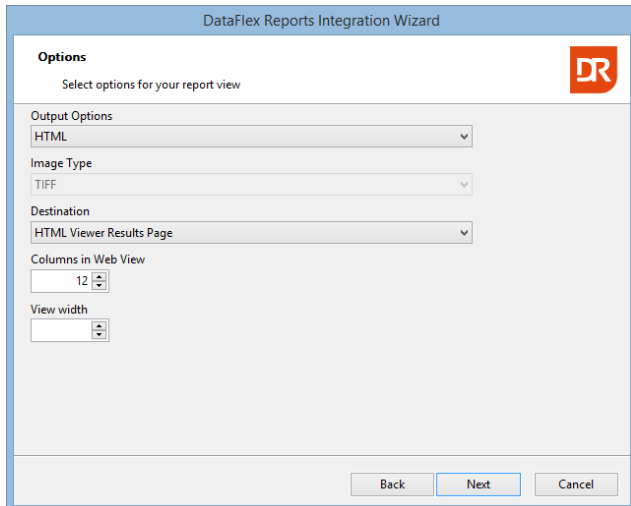
The wizard will create one or two web components based on the choices made during this wizard session. In our case one component will be created and you have to specify (or simply accept) the names for the objects and component filenames. As suggested name, the report name is appended with 'ReportView'.

On the next wizard page you can select a language for localized strings in the report and – if the report is based on ODBC – elect if you want a routine to be written out to change the ODBC connection at runtime.

Finish the wizard and compile / run the web application. You will find the report under 'Views'. You can re-arrange reports under an own menu item labeled 'Reports' if you like.

![DataFlex logo]
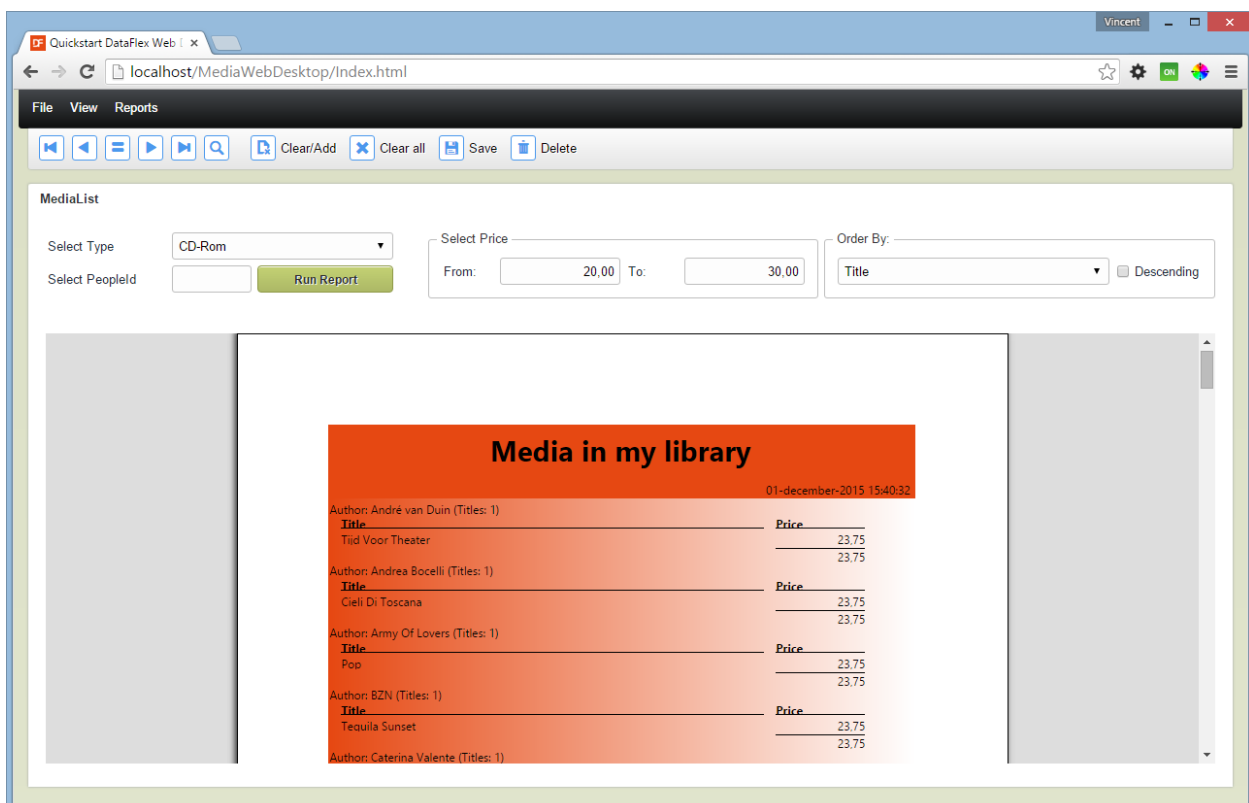
Tip: Read one or more of the blogs about DataFlex Reports integration at the Data Access web-site (http://support.dataaccess.com/Forums).

Get Started!

This concludes this Quick Introduction Guide for DataFlex.

To learn more about DataFlex, visit the following sites to learn more about the product and its creator:

·   www.dataaccess.com
·   www.dataflexcm.com
·   www.visualdatapump.com

Other related sites:

·   support.dataaccess.com/forums
·   www.dynamicai.eu (Business Intelligence tool)

We Look Forward to Helping You to Get Started With DataFlex!

If the documentation, help-files or the forums don't provide you with answers, feel free to ask for assistance via e-mail. Visit the Data Access website for the support options in your region.

Another very good resource is an extensive training guide called "Discovering DataFlex" that contains more than 600 pages. This book has been made available for each revision of DataFlex since the beginning of 2008. Please contact your Data Access sales representative if you would like to purchase a copy of this book. The book is available in PDF format.