

# Welcome to DataFlex 19.0 for Web (Mobile/Touch)

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](http://www.download.com) or [www.dataaccess.com](http://www.dataaccess.com) if you did not yet install the DataFlex Studio and install the product. Use the DataFlexWebAppCheck tool to test if your system is properly configured 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 the new DrillDown - Mobile/Touch style. There is a different Quickstart guide available for desktop styled applications.

We will add a report in a second step. Most of the functionality will be limited by the 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

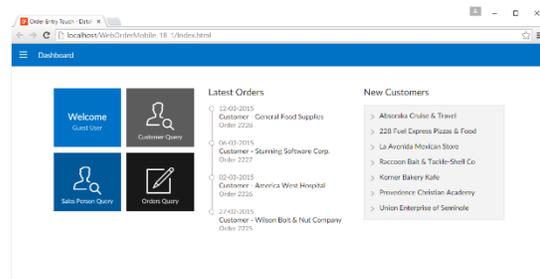
DataFlex is an object oriented programming language. This means that the product – out of the box – delivers classes for all common components. This class is the definition of how such a component looks and functions and what it exactly does once it is instantiated as an object in a written program.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

## 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'. Use the Workspace Explorer in the DataFlex Studio to view the workspace and project information. You can also see a workspace's directory structure in the Configure Workspace Properties dialog.

## DrillDown - Mobile/Touch style

The DataFlex Web Application Framework is highly CSS (Cascading Style Sheet) based for the layout. Besides regular desktop styled web applications, DataFlex now supports Mobile styled web applications optimized for use on mobile and touch devices such as smartphones and tablets. A dashboard with tiles, bigger controls (like buttons) and a responsive design characterizes this style.



In this introduction guide we won't go in CSS itself, which 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.

Later, you can easily convert the tables in the native database to any other supported database backend. Use the DataFlex tools for automatic conversion of existing data and 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.

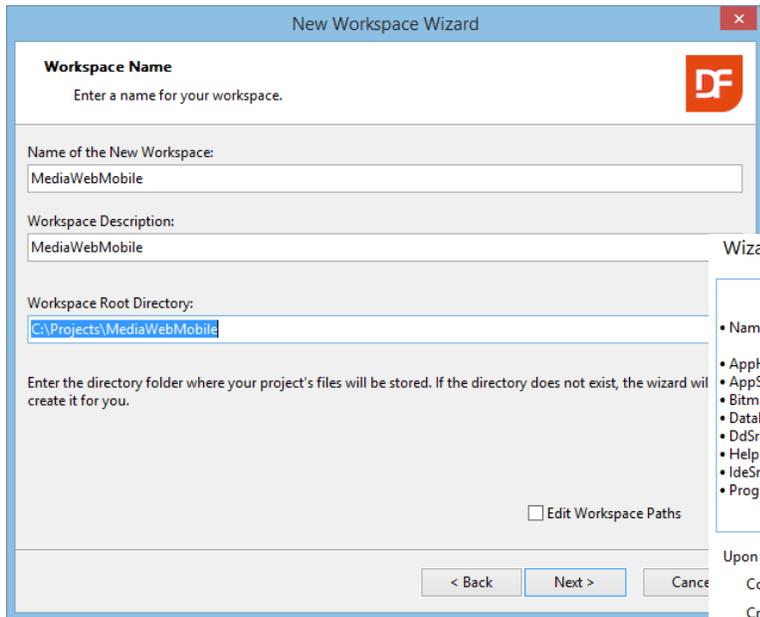
## Our Example Scenario: Media

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 Mobile Zoom View using Wizard to enter data into the People table.  
A web view is a web page to enter data.
- Create the Person Web Mobile Select View to create an overview list of all Person records.
- 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.
  - 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.
  - 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.

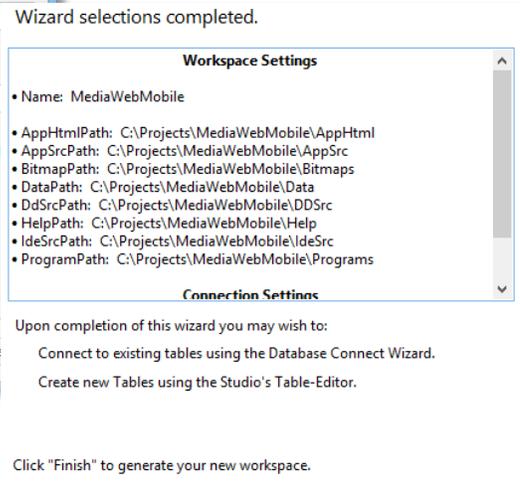
## Getting Started!

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 MediaWebMobile and store it under C:\Projects\MediaWebMobile. 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.

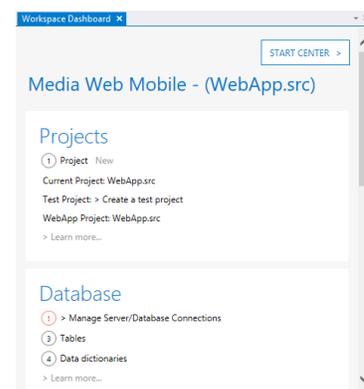
## The Dashboard

The DataFlex Studio opens the workspace dashboard after the workspace has been created. 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 Mobile 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.

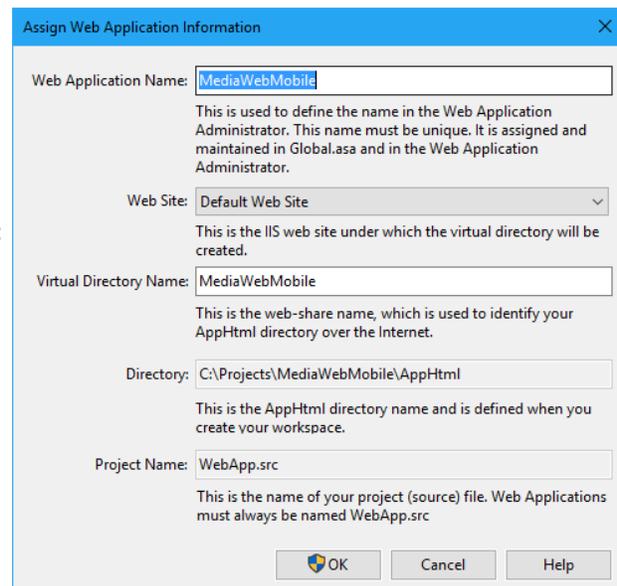
This will open a dialog in which we select "Mobile Web Project". This choice displays a dialog where you then enter the name of the application and virtual directory. The default names values are identical to the workspace name. These names must be unique on the same machine. The virtual directory name,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

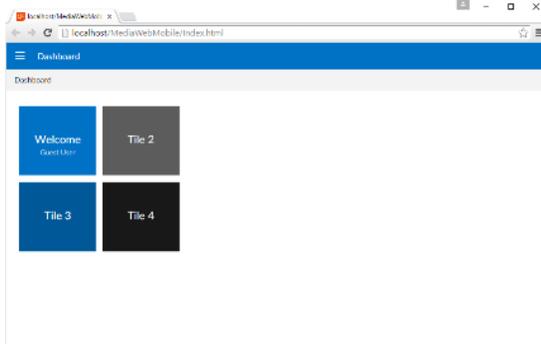
Click OK now; you will see a progress panel showing files being copied. Finally, the Studio creates the WebApp.Src file on disk and makes the MediaWebMobile the current project. The project file contains two main objects one being a cWebApp containing a menu structure. We can already press the **F5** key to run the application.

*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.*

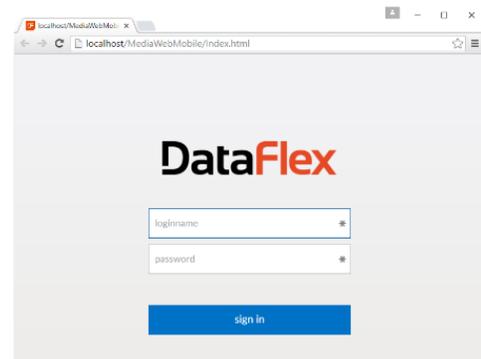


### First run

When run with **F5**, 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.

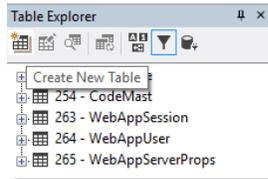


When we first run our Mobile Web project, we are presented with a default dashboard page. This page offers a nice base for navigation through our web application. The tiles can be used as buttons to quickly navigate to the most used pages. For now, we will leave this page be, and focus on the creation of a table in which we will store our data.

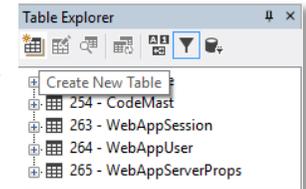


## Create the People Table

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.



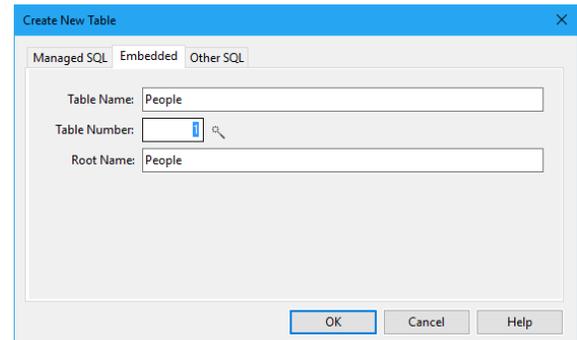
The Table Explorer panel consists of a tool-bar and a list (tree-view) which shows the tables already present in your workspace. A file called filelist.cfg contains the names of the tables. This file is automatically maintained for you.



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.

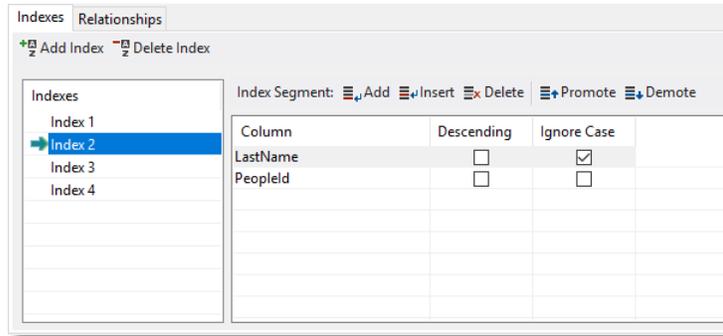
Name	Type	Size	Main Index
PeopleId	Numeric	6,0	
LastName	ASCII	40,0	
FirstName	ASCII	30,0	
Address	ASCII	40,0	
Zip	ASCII	8,0	
City	ASCII	40,0	
Phone	ASCII	25,0	
Comments	Text	1024,0	

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.

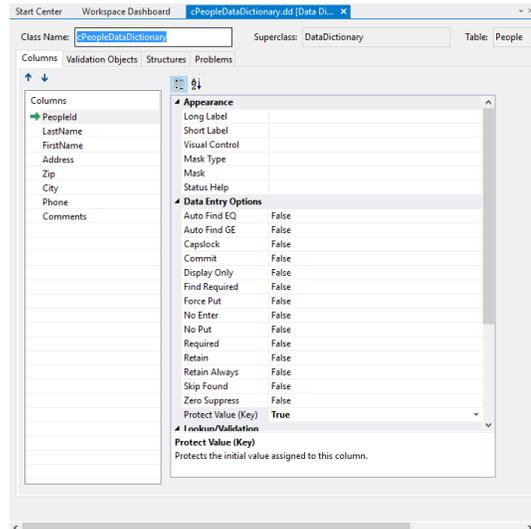
## The Business Rules

When a column is marked as a key-field (Protect value 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.

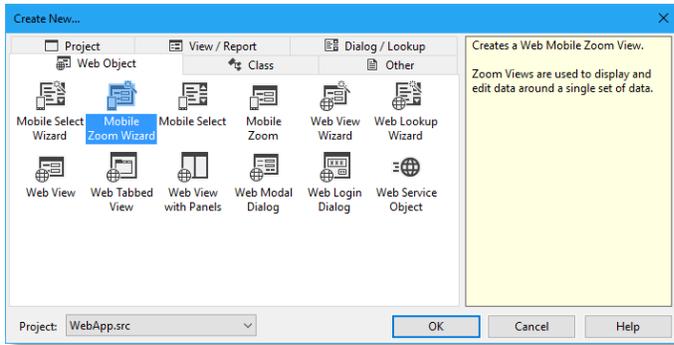
We need to save the table and data dictionary to disk. Either save each one independently, or use the Save all option. If you select the Save all option you also save changed source code which



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 Zoom View

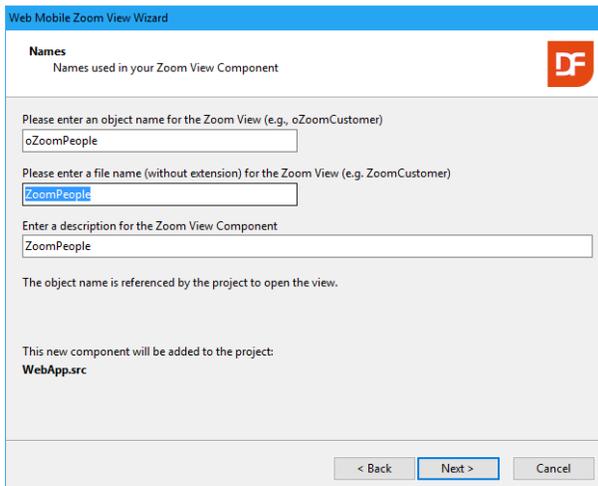
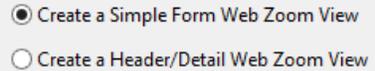
Let us continue creating the application; we will create a data entry view and use the Mobile Zoom Wizard for this. Click on File, New and now Web Object and the following panel appears.



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

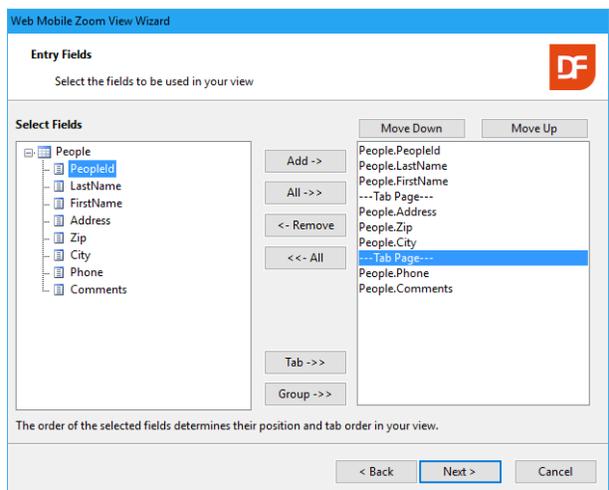
Following the suggested naming conventions in the wizard, enter the following information while processing the wizard:

- The object gets the name oZoomPeople.
- The filename should be ZoomPeople.
- Enter People for the description.
- Create a simple data entry view.
- Choose Simple Form Web View
- Choose the People table



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

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.

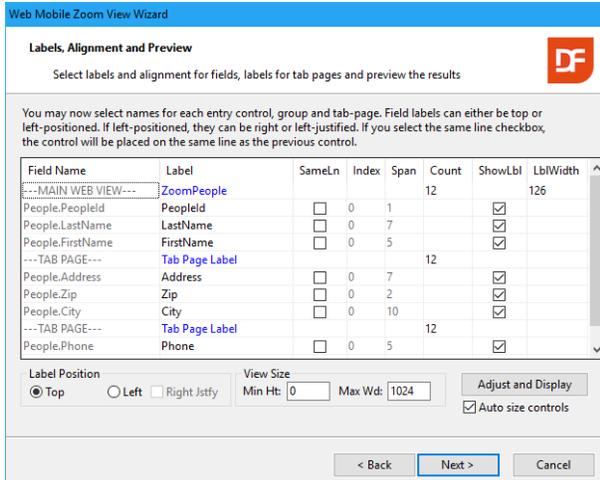


## Module Types

In a DrillDown – Mobile/Touch web project we have two kind of module types.

To view the data in one table row, a 'Mobile Zoom View' needs to be created. Like the name suggests, this type of view, 'zooms in' on a selected table row.

To select a single row, a 'Mobile Select View' needs to be created. This type usually contains a list of rows found in the specified table. For clicking a row in the select view, any action that should be performed can be coded, but the most common operation is to zoom into a row using the mentioned 'Mobile Zoom View' component.

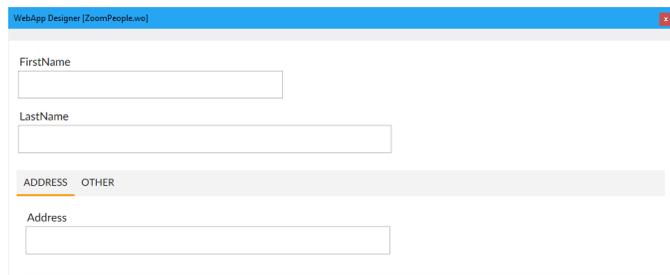


You can click the button labeled "Adjust and Display" to see what your web data entry view would look like. Web controls are laid out in a column based structure (this will be explained in the next chapter named "Layout and Positioning"). The number of columns (automatic or self calculated) determine 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 skip the final step (Assign Parent Lookup Prompt) for now. Finish the wizard. You will be back in the Studio and the result will be loaded automatically. You will see the component source code.

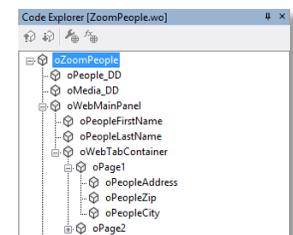
Press **F7** to view the layout in the WebApp Designer. By changing object properties you can change the layout, change labels and more. The object properties are displayed in a panel that can be activated by pressing the **Ctrl+2** key-combination (or via the menu-item View, Properties). If you did 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.



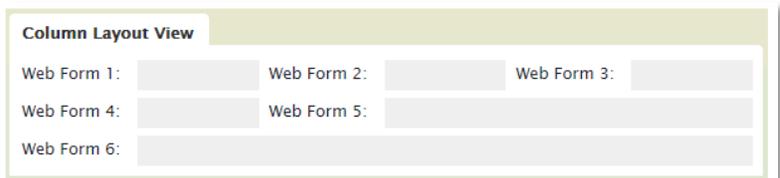
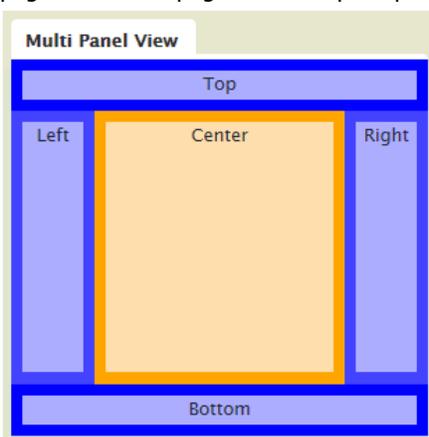
## Layout and Positioning

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 oZoomPeople object in the code explorer window. Notice the view contains one panel (oWebMainPanel) divided into two 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 oZoomPeople there is only one panel and it is the 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



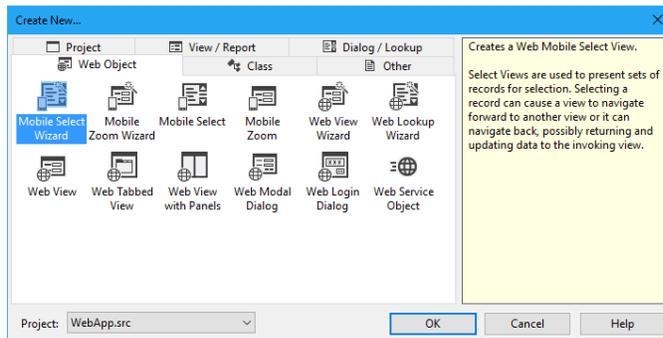
(piColumnIndex) and can span a number of columns (piColumnSpan). If piColumnSpan is set to 0 it tells the system to take all the columns defined in the panel. The first column is column 0. If you want to combine two objects at the same "line", they must divide the available number of columns of the panel 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.

## Creating the People Select View

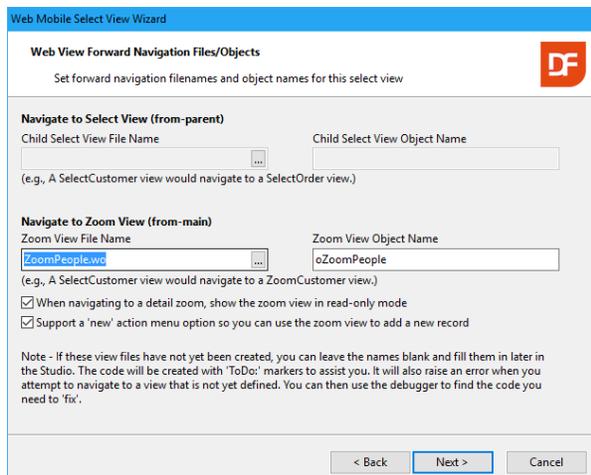
Before we are able to enter data via the Zoom View created in the previous chapter, we have to create a Mobile Select View. A Mobile Select View will display a list containing the rows from the selected table.

We can create this view with the Web Mobile Select View Wizard. Again click on File, New, Web Object and select the 'Web Mobile Select View' icon in the panel.



In the next step, we can select the fields for the view. As this view will show a list of all the rows, we don't want too much data to clutter the list. For now, we will select the LastName, FirstName and City fields, as shown in the screenshot.

When you click 'Next', you can choose how the view will navigate forward when we select a record in the list. Choose Main Zoom, as we do not have a child select view for now.



## Responsiveness

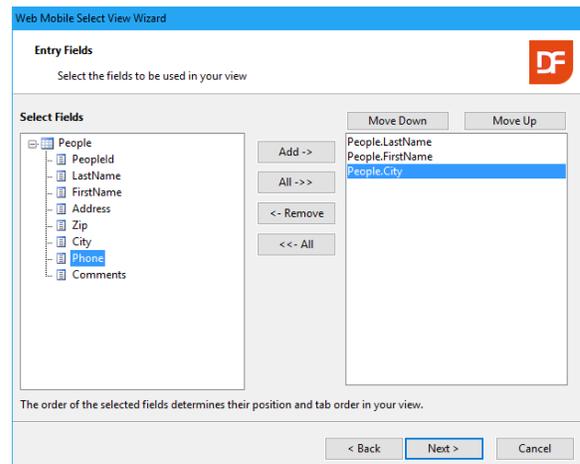
The Mobile/Touch style views are responsive. This means that views and controls will adapt to the screen size of the device they are viewed on.

For example, on a big screen, a form can have multiple controls on one line, where on a small screen (smartphone), the controls each will likely be placed on a separate line, so that all content will fit the screen.

Via the `WebSetResponsive` command you to define how views and controls should respond to different sized devices.

As with the Zoom View, follow the suggested naming conventions in the wizard, enter the following information while processing the wizard:

- The object gets the name `oSelectPeople`, the filename is `SelectPeople` and the description `People`
- Choose the `People` table



Now, we can determine where to the select view will zoom into. In the Zoom View File Name field we can click the three dotted icon which will open a File Browser window. In this window, select the `ZoomPeople.wo` file, which is the view we have created earlier.

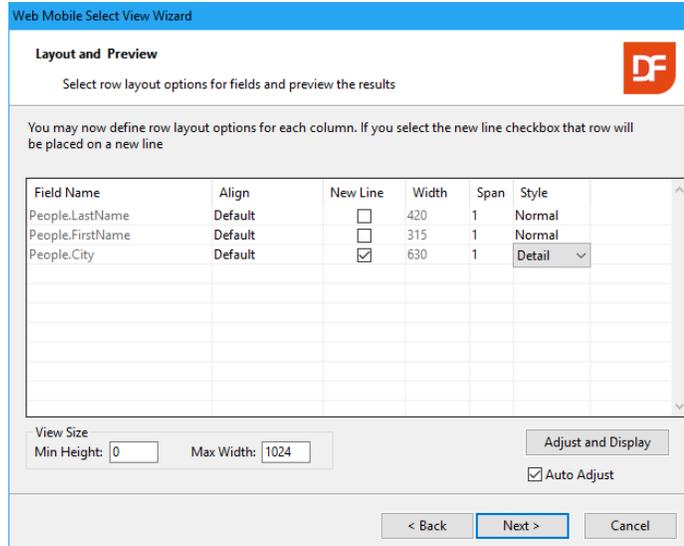
The wizard reads the name of the web view object and shows this under 'Zoom View Object Name'.

In the next step, we can adjust the layout and alignment of the list columns. Check the New Line checkbox for the City field, and change its style to Detail.

The style change will show the city as a greyed out detail in the row.

The New Line option increased the list row height, which makes touch navigation easier.

You can preview this using the Adjust and Display button.



## WebApp Designer Panel

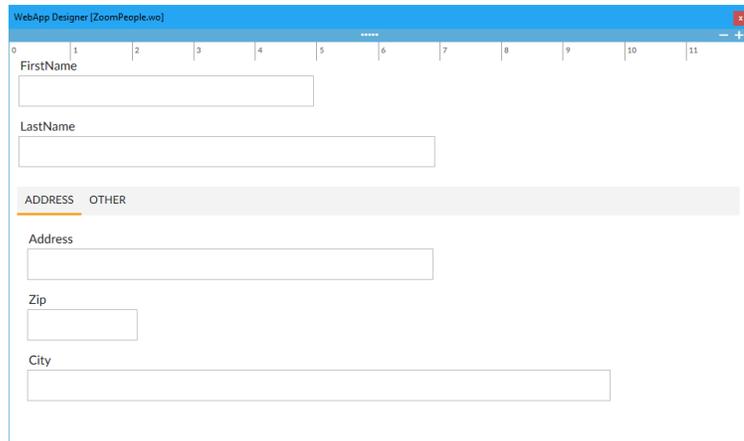
After you have finished the wizard, the Studio shows your Select View in the code editor as well as in WebApp Designer. Press the **F7** key if the designer is not present. Toggle the designer's visible state on and off via this key.

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, it is possible to drag the control 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.

As mentioned before, the application starts with a login box in which you can enter "guest" for the login name and the password.

 Dashboard In the top left corner of the screen you'll see three stacked bars. This icon

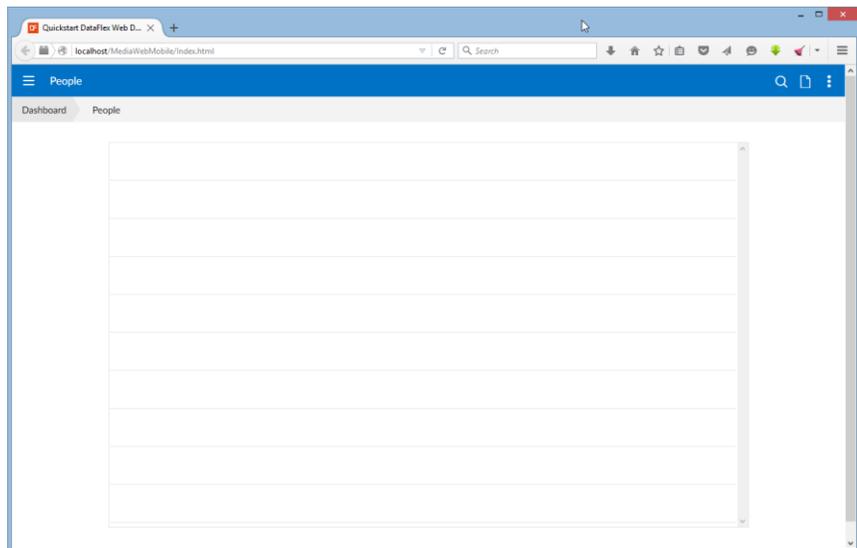
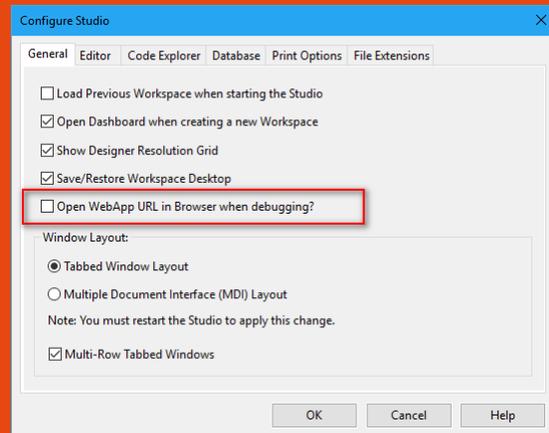
represents the hamburger menu. This is the main menu of the application, where all navigation options will be added to. You can bring up the menu by clicking the hamburger icon.

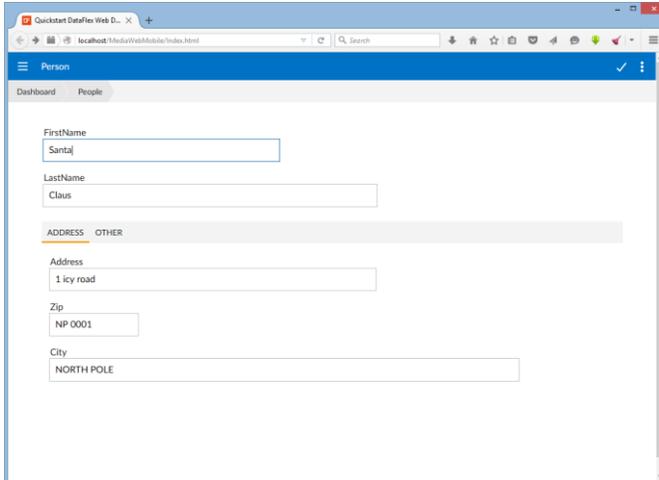
In this menu, we can select the People Select View by clicking on Views and then People. This will show a list of all rows in the People table.

If there were rows with People names in the list you could click a row in the list and navigate to the Zoom View for the selected People row.

Since the People table does not contain any data yet you will see an empty list. You can add new People rows by either clicking the three dots in the top right corner and selecting 'New' or press the 'New' icon (first icon left of the three dots). The icons ('New' and 'Search') and the menu items are called an action group and they can be customized per view. The wizard generated the current action group in the People Select View.

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 be picked up by the browser.





For now; click the 'New' icon which navigates into the People Zoom view. Because the desired action is creation of a new People row the zoom view will not show any data and all input fields are enabled so that a new People row can be created.

Enter the name of a person. The screenshot shows Santa Claus living at the North Pole. After entering the details

If the zoom view was opened by selecting a People row the view would show the details of the People row in read-only mode and you cannot edit the data in the fields until you explicitly choose for 'Edit'.

Opening in read-only mode is the default, because

When navigating to a detail zoom, show the zoom view in read-only mode

on mobile/touch devices, data viewing is more common than data entry. This read-only mode can be turned off in the Web Mobile Select View wizard. To edit the data you can click the edit icon in the top right of the screen, next to the three dots of the action menu. When in edit mode, you can find the following options in this menu:

- Save: Saves the changes made.
- Delete: Deletes the currently selected row
- Clear/Add: Refinds the currently selected row, all changes are dropped

Another component worth mentioning is the breadcrumb trail just beneath the blue menu bar. This will show you where you are, and which drill down options you have chosen. With each breadcrumb you can easily navigate back to the select view, or the dashboard. The text displayed on the breadcrumb can be changed from source code. We will tell you later how to do this.

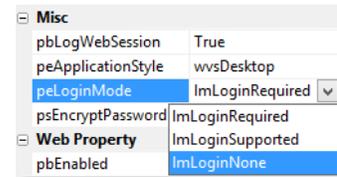
Close the browser application to return to the Studio. Note that sometimes closing the debugger does not close the run mode of the Studio. You can stop the running mode by clicking the stop button.



## 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 management 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 ImLoginRequired to ImLoginNone. For applications that offer additional functionality after logging in the ImLoginSupported option is available.



## Dashboard Tile Link to Login Page

The Dashboard, by default, shows the currently logged in user in the 1<sup>st</sup> tile. We can enhance this tile to change user information or to go back to the login page. To go to the login page use the following steps:

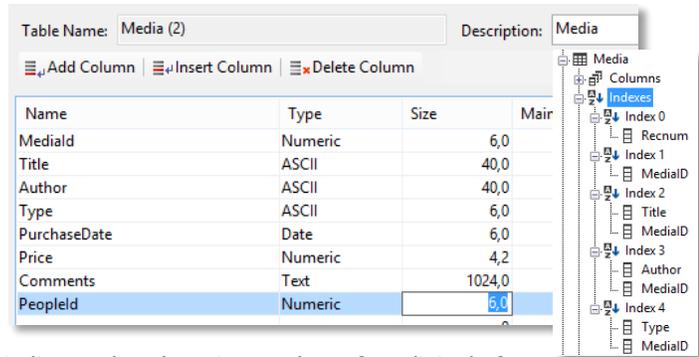
- Set the pbServerOnClick of the oWelcomeTile to true.
- Add the OnClick event via the events tab-page in the object properties.
- Write "Send NavigateBegin of oLoginDialog Self True" in the OnClick procedure
- Insert data-ServerOnClick="LoginAgain" in the first div element of the HTML code of the UpdateHTML message in the OnLoad event

## 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:

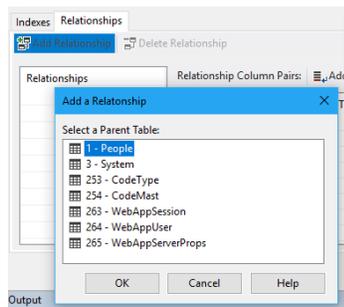
- In the column Type we want to keep track of if it's a CD, DVD, BOOK etc.
- The PurchaseDate is of the type Date.
- The Price is numeric with the 4,2 format. This indicates that the price can have four digits before and two after the decimal point.
- We will use the column PeopleID to connect Media 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 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.



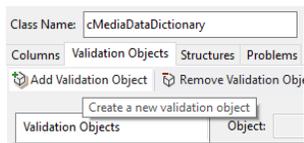
The selection of the parent table opens the option to specify from which child column(s) to which parent column(s) the relationship is made. The type and



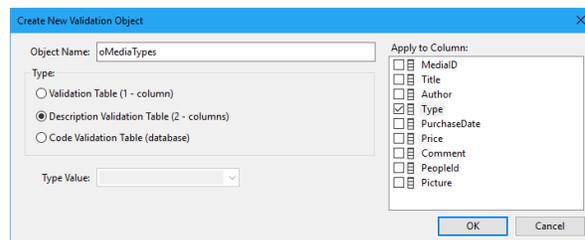
length of the related columns must match and the parent column (usually the key-field) must be uniquely indexed. The current table layout meets these criteria.

## 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.

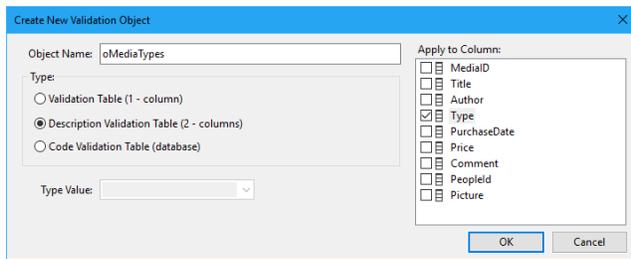


The values for Type should always be entered in uppercase (the Capslock attribute needs to be selected), but let's add something extra. We want the user to use consistent naming when entering Media Types. Enter 'CD' if it is a CD-Rom, enter 'BOOK' if it is a book. Not consistently entering such details 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.

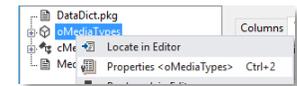
After you clicked the OK button you can start entering values for the table. We suggest you enter the values as shown.



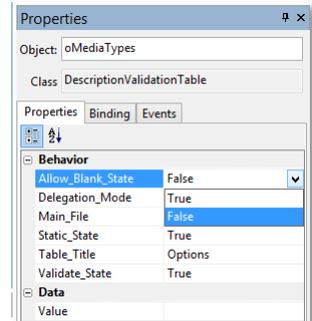
you enter the values as shown.

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.*

### 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.

## Creating the Media Mobile Views

### Media Mobile Zoom View

For viewing and editing Media records, we will need a Media Mobile Zoom View and a Media Mobile Select View. First, we will create the Zoom View, just as we did with the People table. This time the wizard will not be used to create the view. This means you will learn how to make the view in a more manual way. From the menu under File, choose New, Web Object, but now click on: Mobile Zoom.

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

*Tip: The dialog shows that the component will be added to the project WebApp.Src.*

To continue. You are now looking at a very basic cWebView in the Studio that contains some containers, dummy input controls and some comment. The first thing we need to do is to decide which tables we want to maintain in this View.

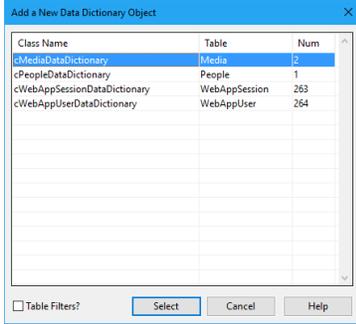
In the menu, under View, open DDO Explorer (or press **Ctrl+4**).



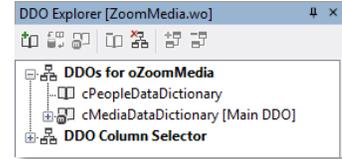
As you can see the DDO Explorer shows no selected tables. Click the "Add DDO" button to adding a DDO. You can do the same from the floating menu (right click on "DDOs for..").



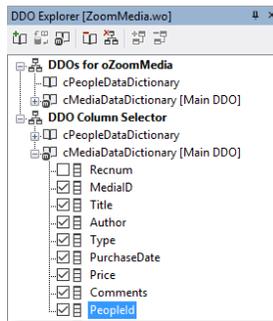
In the dialog that opens, you have to select the right data dictionary for this new view. Since we want to edit (create, modify, delete) the table Media, select the cMediaDataDictionary class (highlighted in the picture).



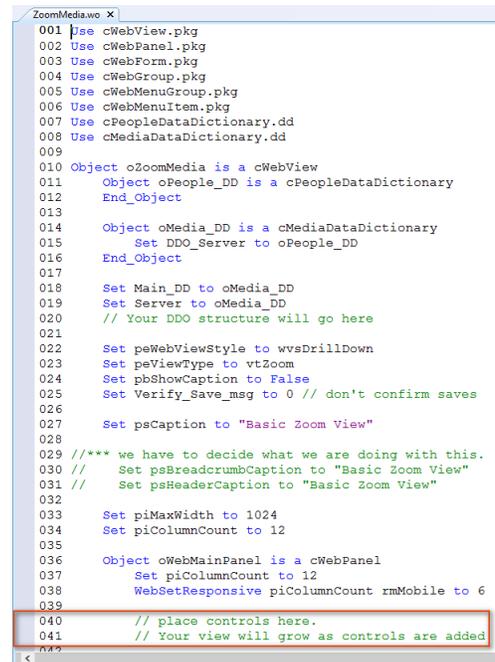
The Studio automatically makes the DDO for the Media table the main DDO and because Media has a relationship to a parent table (People), the DDO for that table will also be automatically added. When the choices are not correct, you can change these via the floating menu on each of the DDOs and apply 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. If inserting them in the source code drop at the location shown in the picture marked with the red outline.



Similarly, drag & drop the column LastName from cPeopleDataDictionary onto the source just after the "PeopleId" cWebForm object just created during the first drag & drop.



The dummy input controls and groups in the panel should be deleted.

### 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. As explained in the 'Layout and Positioning' chapter each container is divided into columns. 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.

After changing the piColumnSpan the LastName object does not move yet as its piColumnIndex (is now 0) needs to be changed to the same value as the piColumnSpan of the oMedia\_PeopleId object. The result will be as shown to the right.

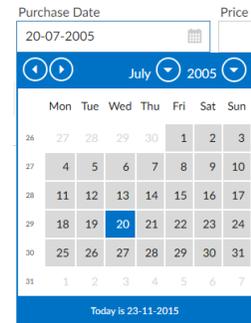


### Date Selector

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.

### Display the Album Cover Image

Likewise, we can change the oMedia\_Picture object to show an image instead of a filename. Locate the object in the source code and change the classname to cWebImage. Change the code into the following:

```
Object oMedia_Picture is a cWebImage
    Set piColumnSpan to 3
    Set pePosition to wiCover
```

Now the control needs to be told how to grab the image stored on disk at the server and transfer it to the client. Each row from the Media table may have an image or not. Technically the images can be loaded from a Web public accessible folder (sub folder of AppHTML) or from a folder that is only available by the DataFlex Web Application. In this paragraph, we will explain the use of the take the private folder location. Let us name the folder AlbumCovers and make it a sibling to AppSrc, Data etc. To retrieve the folder location we will create a method (a function) in the oApplication object (can be found in the WebApp.src file). Add the following function to the oApplication object:

```
Object oApplication is a cApplication
    Function AlbumCoverFolder Returns String
        Handle hoWorkspace
        String sFolder

        Get phoWorkspace to hoWorkspace
        Get psHome of hoWorkspace to sFolder
        If (Right (sFolder, 1) <> "\") Begin
            Move (sFolder - "\") to sFolder
        End
        Move (sFolder - "AlbumCovers") to sFolder

        Function_Return sFolder
    End_Function
End_Object
```

To be able to test the display create a folder named AlbumCovers in the root of the workspace and add a couple of images. Use – for now – Database Explorer (or the Table Viewer) to connect a Media row to an image by copying in the name of the image file in the column Picture.

To show the image stored in the data the following method (DisplayImage) needs to be added to the oMedia\_Picture object.

```

Procedure DisplayImage
  String sFileName sFolder

  Move (Trim (Media.Picture)) to sFileName
  If (sFileName <> "") Begin
    Get AlbumCoverFolder of ghoApplication to sFolder
    Move (sFolder - "\" - sFileName) to sFileName
    Send UpdateLocalImage sFileName
  End
  Else Begin
    WebSet psURL to "about:blank"
  End
End_Procedure

```

This method needs to be called from OnNavigateForward event of the oZoomMedia component. In that event add:

```
Send DisplayImage of oMedia_Picture
```

The UpdateLocalImage method converts the server file path into a download link that is only accessible from within the current Web Application Session, which means that copying the URL (for example into an e-mail) will not work.

Register the download folder as a valid download folder by sending a RegisterDownloadFolder message to the resource manager object. Add the following method to the oWebApp object:

```

Procedure RegisterAlbumCoversFolder
  String sFolder

  Get AlbumCoverFolder Of ghoApplication to sFolder
  Send RegisterDownloadFolder of ghoWebResourceManager sFolder
End_Procedure

```

Then call the method prior to the start of the Web Application. Change the last lines of WebApp.Src to:

```

End_Object

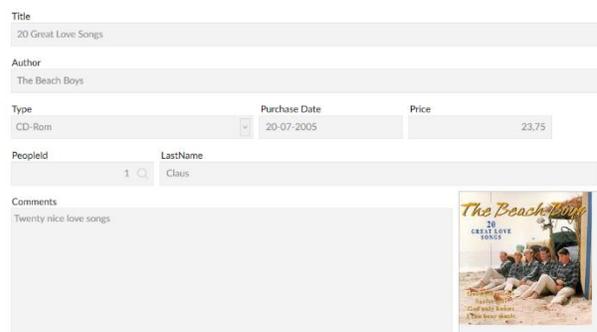
Send RegisterAlbumCoversFolder of oWebApp
Send StartWebApp of oWebApp

```

### 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.

### Person select view for Media

When we want to add a person to a media row, we can fill in the `PeopleID` field, but of course, we won't know all the corresponding ID's for every Person. We want to be able to look up the Person. We can do that by adding the `SelectPeople` view to the `PeopleID` form as a Prompt.

1. Open your `MediaZoom` view
2. Locate the `oMedia_PeopleId` object in the code explorer
3. When selected, change its property `pbPromptButton` to true
4. In the object, add the following procedure:

```
Procedure OnPrompt
Send NavigateForward of oSelectPeople Self
End_Procedure
```

This will make a lookup icon appear in the `PeopleID` form and if clicked, make it navigate to the `SelectPeople` view. The 'self' part of the instruction is the object handle for the invoking object. This defines which object to return to if when closing the `SelectPeople` view. In this case, it will be the current object/view. Use the word 'self' to reference the current object.

### Media Mobile Select View

Now, we also have to create a Web Mobile Select View for Media. Again, from the menu under File, choose New, Web Object, but now click on Web Mobile Select. Name this view '`SelectMedia`'.

As with the just created Zoom View, add the `Media DataDictionary`, and add the `Title` and `Author` fields to the `oWeblist` object. Delete the dummy column object, named `oColumn`, created by the template.

We now have two separate views for Media records, a select view and a zoom view. We have to make some code changes to zoom in on a Media record from the Media Select View. To do this, follow the next instructions:

Locate the `oList` object in the code explorer and inspect its code in the code editor. "There are two procedures already created in this object (`OnRowClick` and `OnGetNavigateForwardData`) which deserve a closer look.

Via the `OnRowClick` event (procedure) we can define what will happen when we click the row. There are several cases defined here by default, which determine from where you have navigated. In this case, our starting point is the main menu, which is not one of the 'nfFrom' navigation types and so we must implement the case else statement. For more information on this, see the help article 'The Forward Navigation Process'.

Passing data to the destination component upon navigation is possible through the `OnGetNavigateForwardData` event. Information like the behavior in relationship to this view.

Replace the code in the Case Else statement of this procedure so it will be as follows:

```
Procedure OnRowClick String sRowID
    tWebNavigateData NavigateData

    Get GetNavigateData to NavigateData
    Case Begin
        Case (NavigateData.eNavigateType=nfFromParent)
            //not used
            Case Break
        Case (NavigateData.eNavigateType=nfFromChild)
            //not used
            Case Break
        Case (NavigateData.eNavigateType=nfFromMain)
            //not used
            Case Break
        Case Else
            Register_Object oZoomMedia
            Send NavigateForward of oZoomMedia Self
    Case End
End_Procedure
```

When you compile the project you can now navigate from the select-view to the zoom-view when you click a row!

The created select view contains a button labeled 'New' but where this automatically worked in the `SelectPeople` view we need to add code in the `SelectMedia` as this component is created via a template and the template does not know or ask if the select-view should navigate into a zoom-view. Locate the object named `oNewButton` and correct the `OnClick` event code to:

```
Procedure OnClick
    Register_Object oZoomMedia
    Send NavigateForward to oZoomMedia Self
End_Procedure
```

## Link Dashboard Tiles to Views

Knowing some views are more often used than other views, we can consider to link to the most used views from the dashboard, so that the users do not have to use the hamburger menu. The generated dashboard contains four predefined tiles and we can even extend them quite quickly.

Say we want to add a link to the SelectPeople view, so the user can quickly find a person in the system. For this: open the Dashboard view and locate the object named oTile2 and view its code. To begin, rename the object to oPeopleTile, so that it is easier to see the purpose of this tile. The OnClick procedure contains commented out example code.

1. Uncomment the code by removing the slashes at the beginning of the lines
2. Replace the 'oYourViewName' placeholder by the object name of the view you want to link to. Use oSelectPeople.

The tile must also tell the user where its link will lead. As it is an HTML box we must edit the psHtml property of the object. You can leave most HTML as is, but change the text between the div tags of class 'Tile\_Title'. This will say 'Tile 2'. You can change it to whatever you like, but in this case, 'People' will do. Now, when you compile and run the project, the second dashboard tile will say 'People' and lead you to the SelectPeople view.

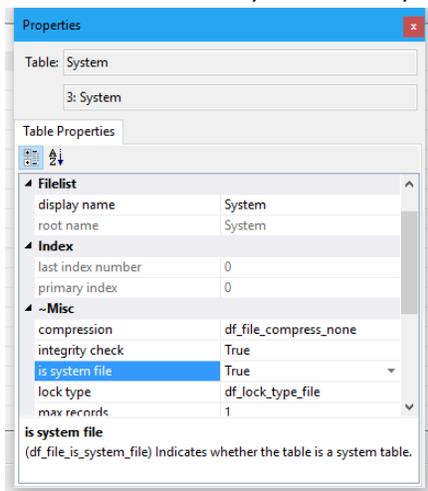
Link the 3<sup>rd</sup> tile to the Select Media view based on above information.

## 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. It only takes two steps to take care of this:

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.

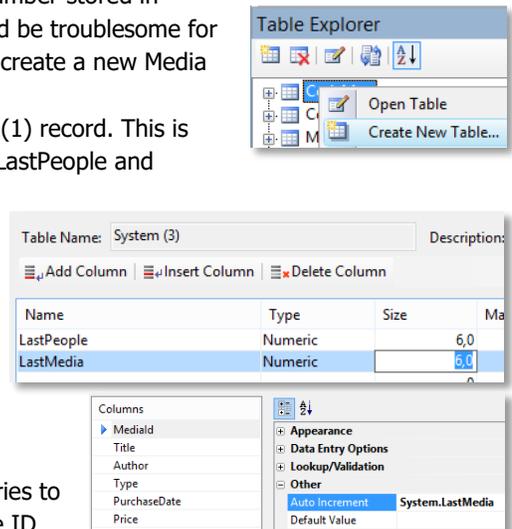
To make sure it actually becomes a system-file, select True for the "is system file" table attribute. Save the table structure before continuing.



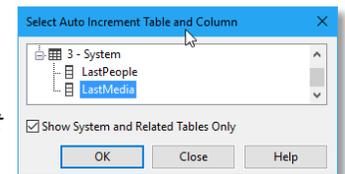
"is system file" table attribute. Save the table structure before continuing.

We want the Data Dictionaries to automatically increment the ID each time a new record is saved.

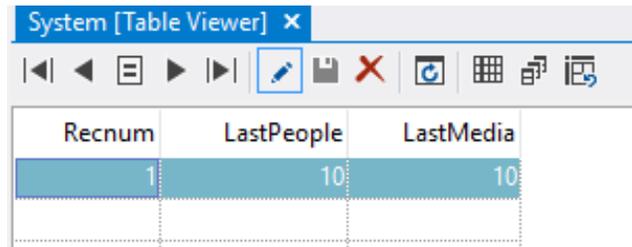
Open the data dictionaries for the tables People and Media in the Studio. Look for the attribute Auto Increment (grouped under Other) in the list of properties for the columns PeopleID (in People) and MediaID (in Media) and click the prompt button. From the list of tables, select System and from the columns the correct source data column – those are LastPeople for PeopleID and LastMedia for MediaID.



*Note: DataFlex developers use a system or a parent table to supply the next ID. That is why you see the checkbox labeled "Show System and Related Tables Only".*



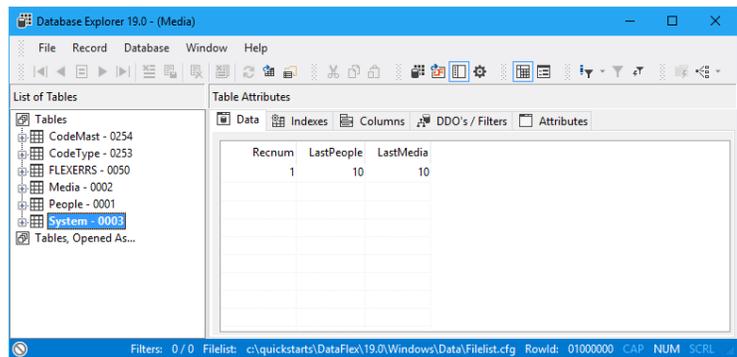
Ok. This should work, if not it is because you have already created some records, with ID's 1, 2, 3 etc. The first time we will use our automated increment function it will try save a record with ID of '1' and that won't work because that value already exists. ID's should always be unique – it is a key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

Set the AutoFind EQ and the NoPut attributes 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.
- 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.
- The framework automatically disables the data-entry in the MediaID and PeopleID controls through setting of the AutoFind EQ and/or the NoPut attributes when navigating into a zoom view. Setting only the AutoFind EQ

attribute disables the controls when navigating into a zoom view with an existing Media row while setting NoPut to true disables the controls also when creating a new Media row.

- 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.

## Show All Media Borrowed

Let us do something a bit more advanced. So far we created select views to navigate into a zoom view but wouldn't it be great if we can view all media that borrowed by a person? Instead of creating a new select and zoom view we will change the behavior of existing views.

Open the Select People view if it is not still opened. Clicking a row in the list currently navigates to the ZoomPeople view. Change the destination from oZoomPeople to oSelectMedia. This will make the selection a real drill-down selection. If you followed the instructions when creating the select view via the wizard you do not have to change the behavior of the little info button as it already navigates to the People zoom view. If it does not, or if you want to check the code, locate the OnClick event in the oDetailButton object and either implement a forward navigation to the oZoomPeople view or check if it already does this.

Compile your project and open the People select view in the browser. Now, when clicking the detail button, the list of media borrowed by the selected person will appear in the list. Clicking the little info button brings you to the details of the selected person.

## Breadcrumb Information

By default the breadcrumb shows the value of the psCaption property of the view. Because the select Media view can now show all rows from the Media table or only the rows connected to a selected person it makes sense to change the breadcrumb caption dynamically. We can do this in the OnNavigateForward event. The view can be opened from the hamburger menu, a dashboard tile or from the People select view. In first two cases the navigation type is nfUndefined, when opened from the People select view the navigation type is nfFromParent. Let's use this information to change the value shown in the breadcrumb. Change the OnNavigateForward event to:

```
Procedure OnNavigateForward tWebNavigateData NavigateData Integer hoInvokingView ;
    Integer hoInvokingObject

    Case Begin
        Case (NavigateData.eNavigateType=nfFromParent)
            Send SetBreadcrumbCaption ("Media Borrowed by:" * People.FirstName ;
                * People.LastName)
            Case Break
        Case (NavigateData.eNavigateType=nfFromChild)
            // If from child, this is a probably a parent lookup from a Zoom
            Case Break
        Case (NavigateData.eNavigateType=nfFromMain)
            // This is not used much with the drilldown style
            Case Break
        Case Else // must be nfUndefined
            Send SetBreadcrumbCaption "All Media"
            // This may be the start of a drilldown query or this may be used for some kind of
            // custom query. You may have set NavigateData.eViewTask to provide more
            // information about this.
    Case End
End_Procedure
```

The semi-colon usage and comment changes are not needed, they are present 'just' for this Quickstart.

## Show the Album Cover Image While Selecting Media

With a little enhancement we can make the application again more attractive. We will add the Album cover to the Media select view. For this we need to add a column to the list that can show an image. Open the SelectMedia view if it is not opened anymore. Locate the details button column in the WebApp Previewer (press **F7** if the previewer is not active) and delete the column via a right mouse click followed by the option delete.



Now open the class palette, if it is not in one of the docking panes (the default is the left hand side docking pane) press the class palette button in the tool-bar. In the class palette locate the `cWebColumnImage` entry and drag it into the list. Change or set the properties shown in the image on the right to the shown values.

```
Set psCaption to "Picture"
Set piWidth to 80
Set pbFixedWidth to True
Set pbDynamic to True
Set piImageHeight to 62
Set piImageWidth to 62
Set piListRowSpan to 2
Set peAlign to alignCenter
Set pePosition to wiCover
```

The most important property in the list for our implementation is the `pbDynamic` property. By setting this to true we can make sure the list shows a different image per row and album covers are usually different per DVD or CD. With `pbDynamic` to true the control fires the `OnDefineImages` event for each created list row. If you add the following code you will get an image per row.

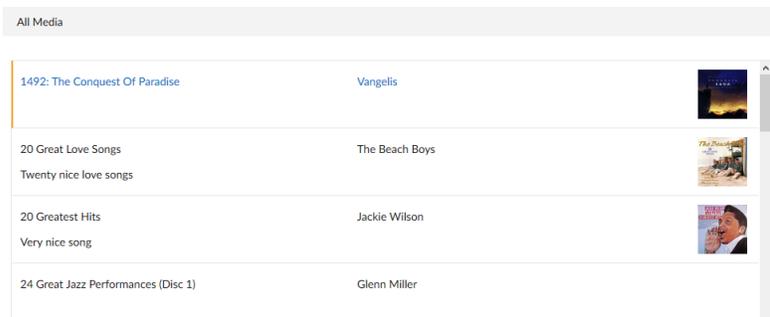
```
Procedure OnDefineImages
  String sFileName sFolder sUrl

  Forward Send OnDefineImages

  Move (Trim (Media.Picture)) to sFileName
  If (sFileName <> "") Begin
    Get AlbumCoverFolder of ghoApplication to sFolder
    Move (sFolder - "\" - sFileName) to sFileName
    Get DownloadURL of ghoWebResourceManager sFileName to sUrl
    Send AddImage sUrl
  End
End_Procedure
```

In the code above the function `AlbumCoverFolder` returns the location of the images folder. First the routine checks if the `Picture` column in the current `Media` row contains an image (file)name and if this is true it is append to a path where the images are stored. The function to return the path was added while enhancing the `Media Zoom` view.

The result of the `Media` select view can be as shown in the picture on the right.



## 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 View`.
- Drop a `cWebForm`, a `cWebButton`, a `cWebCheckbox` and a `cWebList` object. Change the name of the `cWebList` object to `oMediaList`.
- 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!".
- Drag some data columns (Title, Author, Price...) from the DDO Column Selector (in the DDO Explorer) to the cWebList object.

To get the list fill itself automatically when the view opens add the OnNavigateForward event to the view (via the object properties, events tab-page or directly in the code) and inside the event add the following:

```
Send FindFromTop of oMediaList
```

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.

Filter on:   Case Sensitive

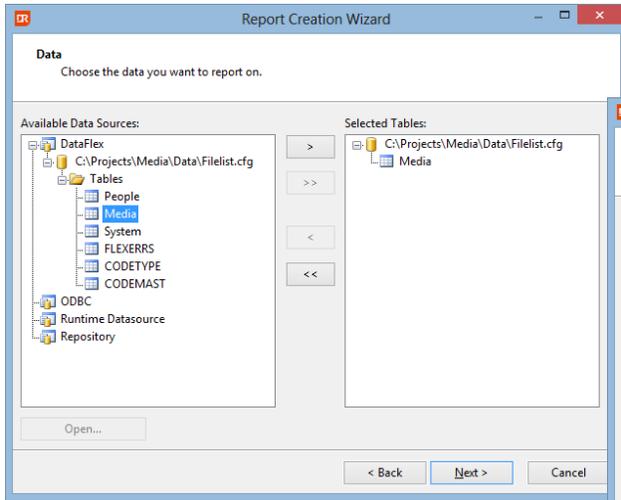
Title	Author	Type	PurchaseDate	Price
Elvis Presley Artist Of The Century (Dis	Elvis Presley	CD	20-07-2005	9,99
Elvis Presley Artist Of The Century [Dis	Elvis Presley	CD	20-07-2005	44,95
Artist Of The Century [Disc 2]	Elvis Presley	CD	20-07-2005	44,95
Best Of The 50's	Elvis Presley	CD	20-09-2001	39,95

## Reports and Lists

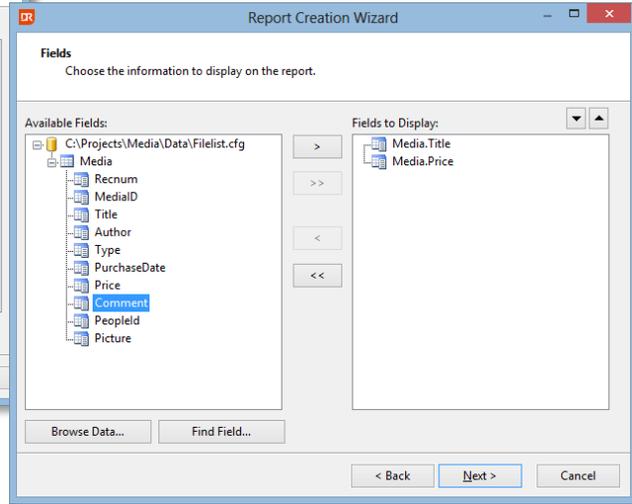
One of the most powerful ways to create reports and integrate them in DataFlex is by using DataFlex Reports and the DataFlex Reports Integration Library. This wizard automatically integrates a report in your Web application. The end-user will be able to start the report from the menu and still be able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

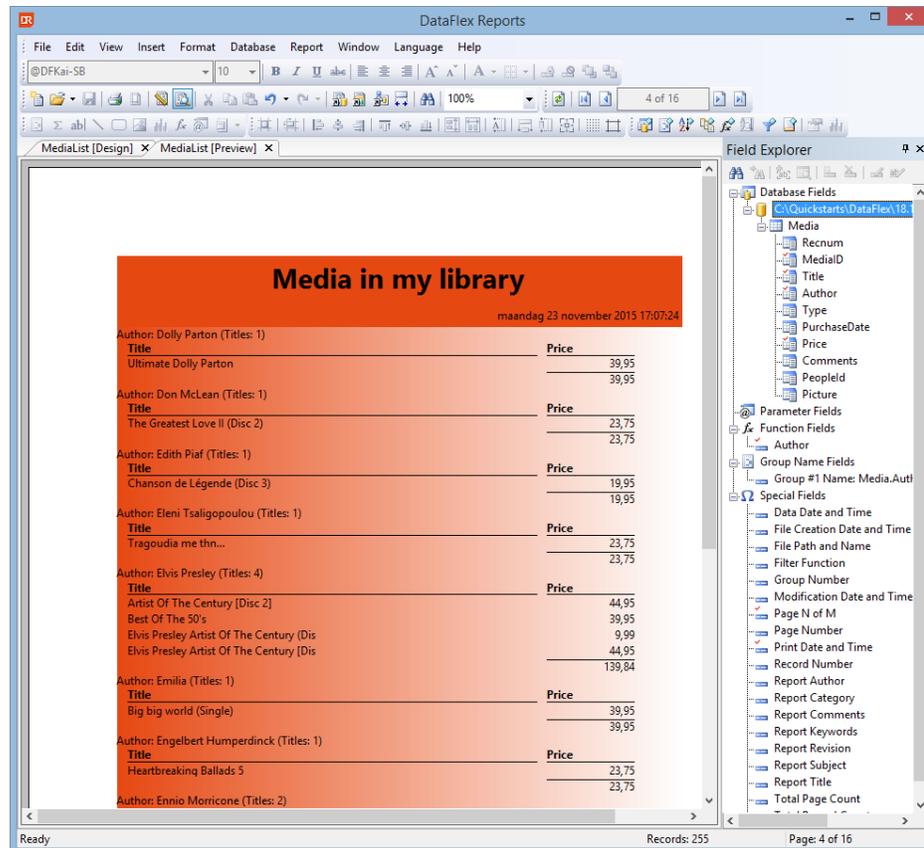
Skip the summary, data filters and repository pages for this report .

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

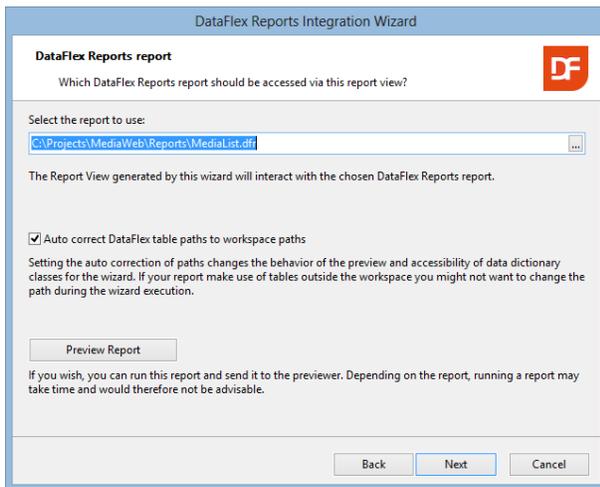


optionally (turned on) installs an integration library. Attach to the library by selecting Tools, Maintain Libraries. In the dialog that pops up you will see the already connected libraries.

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!

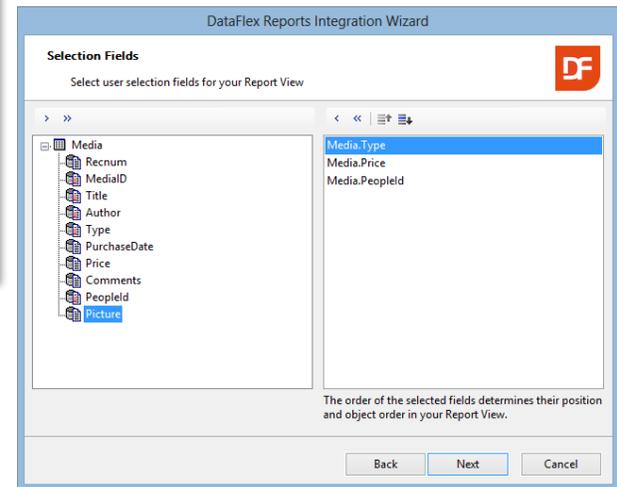
To integrate the report we start the integration wizard. Select File, New, Web Object and pick the DataFlex Reports Integration Wizard. Note that there is also a wizard for Windows programs but you need to one in the Web Object page.

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.

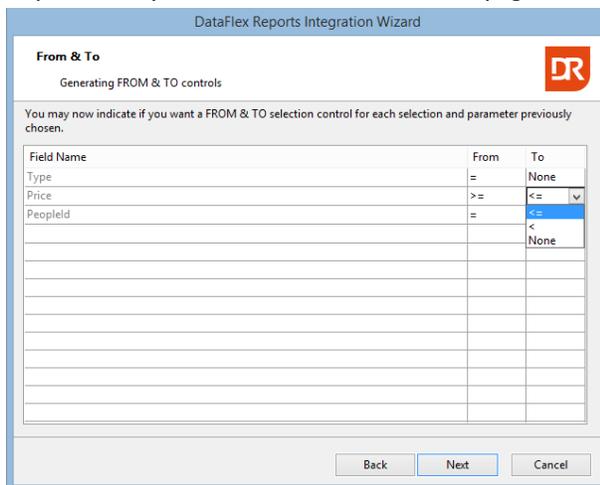


After clicking the 'Next' button you can select / confirm the application style. Use the 'DrillDown – Mobile/Touch' option.

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



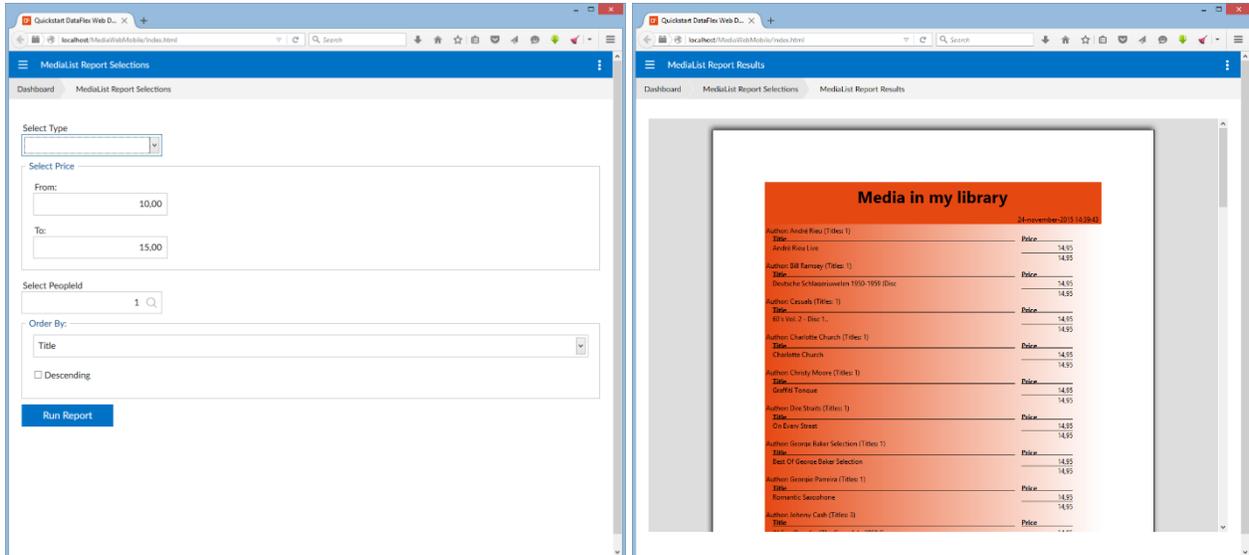
report. It takes too long for the average impatient web user and it also makes no sense to view so many pages over the web. You even might want to check if the user made selections or not.

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.



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.



Tip: Read one or more of the blogs about DataFlex Reports integration at the Data Access web site (<http://support.dataaccess.com/Forums>).

## Picture Selector & Upload Views

If you are still hungry... we can extend the project with a view for selecting from all available album cover images and a view to upload a new cover image to the system.

### Picture Selector View

Go to the Create New dialog in the Studio again and create a Select View. Do not use the wizard as the data will not be read from a database but from the file system. Use `oSelectAlbumCover` as the name of the select view. Select the `cWebList` object and change the name to `oFilesList`. Change the `pbDataAware` property to false. Then create two columns. Name the first column `oFileNameColumn` and base it on the `cWebColumn` class. Name the second `oPictureColumn` and base it on the `cWebColumnImage` class. Set the properties of this image column identical to the image column in the select media view.

To get data (filenames and the image) into the list we need to implement the `OnManualLoadData` event. Add this event to the object via the object properties panel. When the list needs data it fires this event passing the current data via an array. Use the same array to add extra (or changed).

Inside the `OnManualLoadData` routine we start with getting the path to the `AlbumCovers` folder in the workspace. This has been done twice in this Quickstart, look it up.

Filenames can be read from disk via a `DIRECT_INPUT` command passing the path and a device constant `'DIR:.'`. Each directory entry is read with a `READLN` statement. Sub-folder entries can be recognized by the first character of the directory entry; if it is a square bracket it is a folder. This is not a DataFlex invention it is how the operating

```

Command Prompt
C:\Quickstarts\DataFlex\18_1\Web Mobile>dir /w
Volume in drive C has no label
Volume Serial Number is ABF0-88DF

Directory of C:\Quickstarts\DataFlex\18_1\Web Mobile

[.]                [.]                [AlbumCovers]
[AppHtm1]          [AppSrc]           [Btmgs]
[Data]             [DBSrc]            [Help]
[IdeSrc]           [Media Web Mobile.sws] [Programs]
[Reports]

1 File(s)                225 bytes
12 Dir(s)                175.619.293.184 bytes free
C:\Quickstarts\DataFlex\18_1\Web Mobile>

```

system passes the information to the virtual machine (open a Windows CMD window and enter 'DIR /w' if you want to see this).

Each 'valid' directory entry (a filename that we want to have in the list) is used 3 times for each list row.

First the value inclusive the path is used as row identifier. Each list row needs to have a unique row identifier and the file plus path is unique. It also is needed to tell the caller object what file was selected by the user. The 2<sup>nd</sup> use is to show the name of the file to the user and the 3<sup>rd</sup> use is to show the contents of the image file as picture.

Study and copy the following code to get the manual list working:

```

Procedure OnManualLoadData tWebRow[] ByRef aFiles String ByRef sCurrentRowID
    String sFolder sFileName
    Integer iChannel iRow

    Move (Seq_New_Channel ()) to iChannel
    If (iChannel >= 0) Begin
        Get AlbumCoverFolder of ghoApplication to sFolder
        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
                Move (Trim (sFileName)) to sFileName
                Move (sFolder - '\' - sFileName) to aFiles[iRow].sRowID
                Move sFileName to aFiles[iRow].aCells[0].sValue
                Get DownloadURL of ghoWebResourceManager (sFolder - '\' - sFileName) ;
                    to aFiles[iRow].aCells[1].aOptions[0]
                Increment iRow
            End
        End
    Loop
    Close_Input channel iChannel
    Send Seq_Release_Channel iChannel
End
End_Procedure

```

To complete the component change the content of the OnRowClick event to:

```

Procedure OnRowClick String sRowID
    Send NavigateClose Self
End_Procedure

```

Upon return the invoking object fires the message OnGetNavigateDataBack and in this routine we 'simply' pass the current row identifier (remember that it contains the filename inclusive path) to the caller. Change the code of this event to:

```

Procedure OnGetNavigateBackData tWebNavigateData ByRef NavigateData Handle hoBackToView
    Forward Send OnGetNavigateBackData (&NavigateData) hoBackToView
    WebGet psCurrentRowID to NavigateData.sRowID
End_Procedure

```

To get the Picture selector working; change one more thing. We need to implement an event to fill the list. You can choose between OnNavigateForward and OnShow. If you use this selection page only to select an existing image the OnNavigateForward will do but we also want to implement a way to upload new cover images to the system and

when we call that feature from the select view we need to get the list refreshed on return of the upload feature and then OnShow is better. So add:

```
Procedure OnShow
    Send GridRefresh of oFilesList
End_Procedure
```

The event does not fire if we forget to set the pbServerOnShow property of this selection view to true.

The select view should not be started from the hamburger menu or a tile and therefore remove the option from the oViewMenu object in WebApp.src. Of course this means we need to add a way to navigate to the select view. For this add a button (cWebButton) object underneath the cWebImage control in the oZoomMedia view. Add the following code:

```
Object oSelectImageButton is a cWebButton
    Set piColumnIndex to 11
    Set psCSSClass to "WebPromptMenuItem"
    Set psToolTip to "Select an Album Cover Image"

    Procedure OnClick
        Send SelectImage of oMedia_Picture
    End_Procedure
End_Object
```

As you can see the code does not directly navigate to the select album cover image but instead sends a message to the cWebImage object for that. This is done because the image selection consists of two parts; the navigation to the image selector view and code that needs to be executed when returning from the view. Add the following code to the oMedia\_Picture object to select the image.

```
Procedure SelectImage
    Boolean bEnabled

    WebGet pbEnabled to bEnabled
    If (bEnabled) Begin
        Register_Object oSelectAlbumCover
        Send NavigateForward of oSelectAlbumCover Self
    End
End_Procedure

Procedure OnNavigateBack Handle hoCallback tWebNavigateData NavigateData
    Send UpdateLocalImage NavigateData.sRowID
    Set Field_Changed_Value of oMedia_DD Field Media.Picture to ;
        (ExtractFileName (NavigateData.sRowID))
End_Procedure
```

If you want to make it possible to remove an image from an existing media add the following button prior to the oSelectImageButton object:

```
Object oClearImageWebButton is a cWebButton
    Set piColumnIndex to 10
    Set psCSSClass to "WebClearMenuItem"
    Set psToolTip to "Remove the current Album Cover Image"

    Procedure OnClick
        WebSet psURL of oMedia_Picture to "about:blank"
```

```

        Set Field_Changed_Value of oMedia_DD Field Media.Picture to ''
    End_Procedure
End_Object

```

To avoid the two buttons can be clicked when the zoom view is opened in read-only modus (default) add the following code to the SetActionButtons method of the view:

```

WebSet pbEnabled of oClearImageWebButton to (not (NavigateData.bReadOnly))
WebSet pbEnabled of oSelectImageButton to (not (NavigateData.bReadOnly))

```

## Picture Upload View

This feature is easy to implement BUT requires a couple of security measurements. You need to set security rights on the upload folder to the IUSR account and you need to register the upload folder in your web application as a valid upload folder. For the latter one add the following line to the RegisterAlbumCoversFolder method added in the chapter "Display the Album Cover Image":

```

Send RegisterUploadFolder of ghoWebResourceManager sFolder

```

Now we need to create the component for uploading. Create a new Mobile Zoom view and name the view object oUploadAlbumCover. Remove the controls and containers inside the oWebMainPanel object (they are created by the template). Find the cWebFileUploadButton class in the class palette and drag it to the main panel object. Inside the upload button implement the OnFileUpload event to specify the location where the file needs to be uploaded to. Do this with the following code:

```

Function OnFileUpload String sFileName Integer iBytes String sMime Returns String
    String sPath

    If (Left (Lowercase (sMime), Pos ('/', sMime) - 1) = "image") Begin
        Get AlbumCoverFolder of ghoApplication to sPath
        Move (sPath - '\' - sFileName) to sPath
    End

    Function_Return sPath
End_Function

```

Now the user can press the upload button when navigating to this view to select files from the local device and upload to the server. Add code to check the MIME type to avoid uploading a non-image file.

Final operation is to alter the 'New' button in the oSelectAlbumCover object's ActionGroup so that selecting this option brings the user from the selection view to the upload view. Change the 'New' button to:

```

Object oNewButton is a cWebMenuItem
    Set psCaption to C_$New
    Set psCSSClass to "WebClearMenuItem"

    Procedure OnClick
        Register_Object oUploadAlbumCover
        Send NavigateForward to oUploadAlbumCover Self
    End_Procedure
End_Object

```

## 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](http://www.dataaccess.com)
- [www.dataflexcm.com](http://www.dataflexcm.com)
- [www.visualdatapump.com](http://www.visualdatapump.com)

Other related sites:

- [support.dataaccess.com/forums](http://support.dataaccess.com/forums)
- [www.dynamical.eu](http://www.dynamical.eu) (Business Intelligence tool)

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.

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

