

Welcome to DataFlex 19.1 for Web (Mobile/Touch)

DataFlex is a complete software platform for rapidly developing and deploying Windows, web, mobile and cloud business applications - fast! It is one of many products Data Access Worldwide has delivered to the software development community since 1976. DataFlex is for application development and is available as a Commercial Edition, which is used by businesses, institutions and government agencies to build and deploy applications, and as a Personal Edition, which is a free, fully functional edition that can be used for non-commercial, private use. At this point, you should have installed the DataFlex Studio. If not, before you read through this introduction, make sure to download a copy from www.dataaccess.com and install it.

During installation, use the DataFlex WebAppCheck tool to test whether your system is properly configured for creating web applications.

Once you have DataFlex Studio installed, you will be ready to follow through this document to learn the steps involved in building a basic database web application with DataFlex that includes data entry and reporting. This document will focus on the new DrillDown - Mobile/Touch style web applications, but we offer a Quickstart guide Desktop style applications.

Once you get the basics down, get further into the underlying programming language and framework to take advantage of the whole range of features that DataFlex has to offer.

First, let us start by covering some concepts important to fully understand the DataFlex framework.

Object oriented

DataFlex is an object oriented programming (OOP) language. This means that all common components belong to a pre-defined class. The classes hold the definition of how such components look, function, and what they exactly do. Components are defined as objects of certain class in your application and that guarantees consistency and reusability of code throughout your project.

A major advantage of using OOP is that a lot of technical details are defined in the classes and you can concentrate on the real functionality of the application you want to develop customizing components as you need.

Workspace

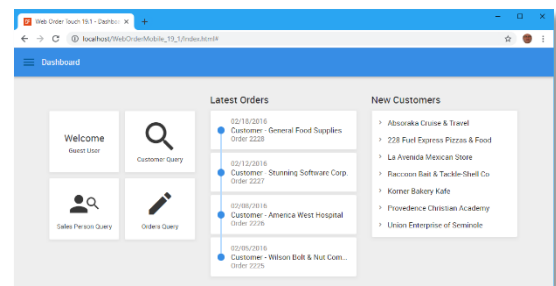
Before you start a project, you need to create a new environment where all the elements of your project – such as programs, objects, and rules – will be stored. That environment is called a *workspace*. A workspace is a set of folders in which the database, source-code and all files necessary to your project are stored.

A workspace can hold one or more projects, which are the eventual programs or 'executables', but only one web project – WebApp.src. In the Studio, Workspace Dashboard gives you an overview of the workspace and more details are displayed in Workspace Explorer and Code Explorer. You can also see a workspace's directory structure in the Configure Workspace Properties dialog under the Tools menu in the Studio.

DrillDown - Mobile/Touch style

The DataFlex Web Application Framework relies on CSS (Cascading Style Sheet) for defining application 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 will not go in CSS itself, which is more an expert level feature.



Database

DataFlex can work with any popular database management system (DBMS). DataFlex comes with the necessary database Connectivity Kits that allow you to utilize those DBMS in your project, but for the purposes of this introduction we will limit ourselves to the embedded DataFlex database, our native database.

Databases are maintained in the DataFlex Studio. The Studio allows for the creation of tables, the definition of business rules and custom coding.

At a later time, you may easily convert native tables to any other supported database backend using the available DataFlex conversion tools that will not only perform the conversion of table structures but also their existing data.

Data Dictionary

Data dictionaries are the business rules in your workspace that keep your data accurate and consistent. For example, before saving records, the entered data needs to be validated – states should be uppercase and of a certain value, and customers may not order more than the value specified in his credit limit.

Those validations are referred to as business rules and the files where they are stored are called data dictionaries. All applications using data dictionaries will follow those same rules and if any rules change, only the data dictionaries need to be adjusted and the new rules will be applied throughout all applications.

Having data dictionaries also means that if you were to migrate the application to another database platform, the same business rules will automatically be applied no matter what database backend is used. Data dictionaries are created and maintained 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. These two tables will be linked to each other in such a way that you can keep track of the location of each item in your media library.

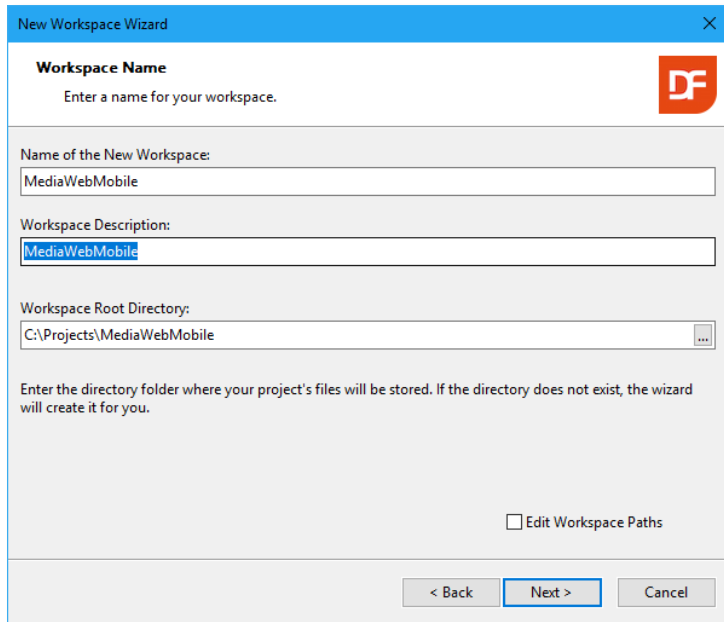
This example will take you through the following steps:

- Create a workspace
- Create a project named Media
 - All components in a workspace must be created while there is a project 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 Web Mobile Zoom View using a 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
 - Ensure consistent format of the way the column Media.Type is entered
 - 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 person in People owns or borrowed what Media item
 - 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. 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. Since Windows has strict rules about the location where web shares are accessible, create the workspace in C:\Projects\MediaWebMobile. 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

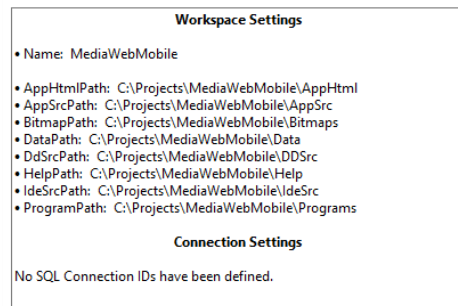


summary of the gathered information and what to do next.

The wizard creates the selected folders for the Media workspace and returns to the Studio so that you may take the next steps.

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

Wizard selections completed.



Upon completion of this wizard you may wish to:
Connect to existing tables using the Database Connect Wizard.
Create new Tables using the Studio's Table-Editor.

Click "Finish" to generate your new workspace.

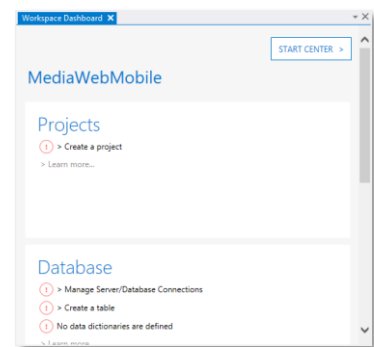
The Dashboard

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

The dashboard as shown here says that the next step is either 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 that are picked up by the dashboard. This means you can use the dashboard as a project management tool.



Create the Mobile Web Project

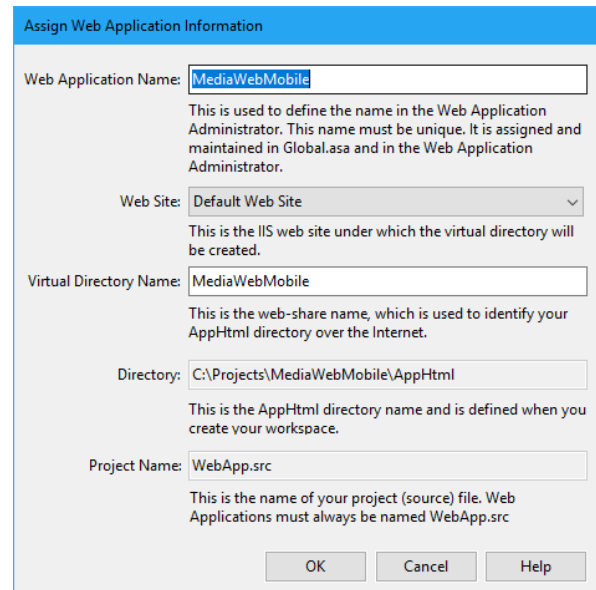
The next step is to create a project. For almost everything we create in the Studio we need an active project. There are several ways in the Studio from which you can create a project. For now, click the option in the dashboard. Alternatively, you can also create a new project via the File pull-down 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. That name cannot be changed.

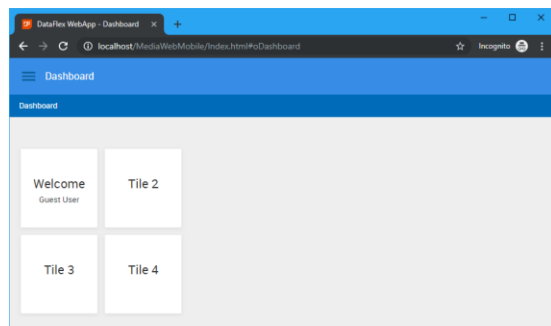
Web applications can run under the default website or under a specific website. Applying running under a specific website is too advanced for this introduction.



Click OK and a progress panel shows the files being copied. Finally, the Studio creates the WebApp.Src file on disk and makes MediaWebMobile the current project. The project file contains two main objects. One of them is a cWebApp that contains a menu structure. At this point we can press the **F5** key to run the application.

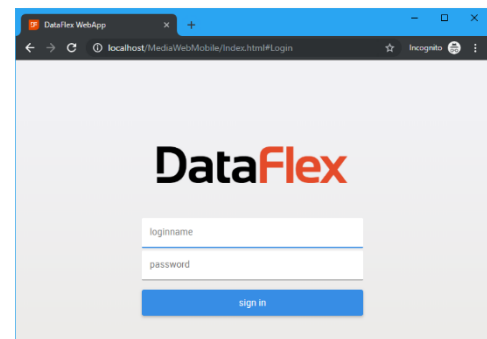
First run

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.



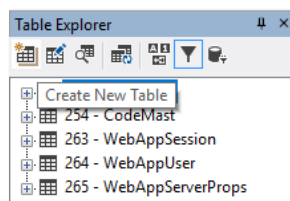
When we first run our Mobile Web project, we

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.



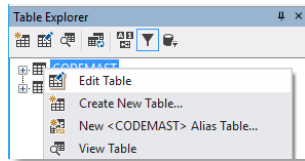
Create the People Table

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.



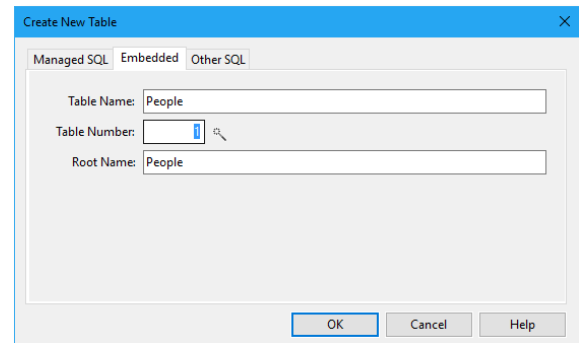
The Table Explorer panel consists of a tool-bar and a list (tree-view) which shows the

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 or edit a table for modification, view the contents of a table, refresh table definition, sorting and/or filtering the list of

tables and as last one opening the SQL Connection Manager. 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.



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 table number value can be changed but the suggestion is good to go.

Press 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.

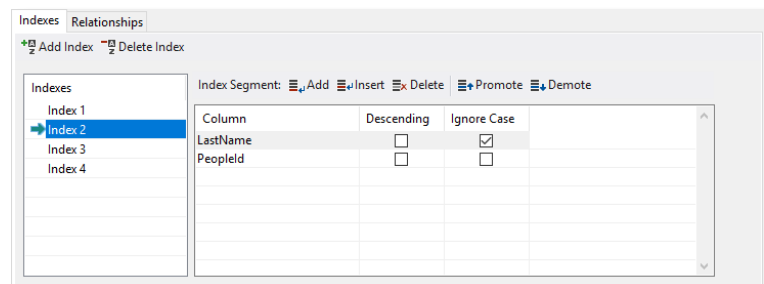
Table Name: People (1)		Description: People	
<div> Add Column Insert Column Delete Column </div>			
Name	Type	Size	Main Index
PeopleId	Numeric	6,0	
LastName	ASCII	40,0	
FirstName	ASCII	30,0	
Address	ASCII	40,0	
Zip	ASCII	8,0	
City	ASCII	40,0	
Phone	ASCII	,0	
Comments	Text	1024,0	

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 people, 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 those columns 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".

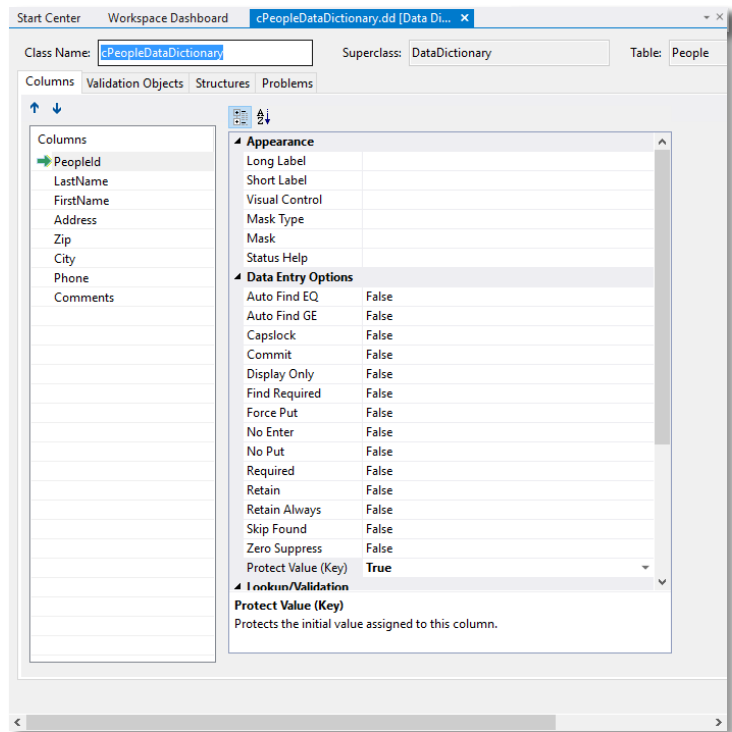


The tab has two toolbars. One - with the buttons "Add Index" and "Delete Index" – and this works on the list of indexes making it possible for you to add or remove a complete index while the other toolbar works on the grid with index segments. Change the order of index segments with the "promote" and "demote" tool-bar buttons if they are in not in the desired order.

Save the table structure for the table `People` by pressing the `Ctrl+S` key-combination or click the Save tool-bar button.

The Business Rules

When a column is marked as a key-field ('Protect value (Key)' attribute) the value cannot be changed after the record has been created. PeopleID is used to link the rows between the later to be created Media table and the 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 the key field will be set.



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 – select Zip from the columns list and change the Capslock attribute to True; then repeat the same steps for City.

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 changes after saving them as long as you do not close the file. Closing a file would affect the undo stack and you may not be able to undo all the changes.

Creating the People Zoom View

We can now create a data entry view and we will do so with the Web Mobile Zoom View Wizard. Again, click on File, New and now Web Object and the Create New panel just like the one pictured on the right will be displayed.

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

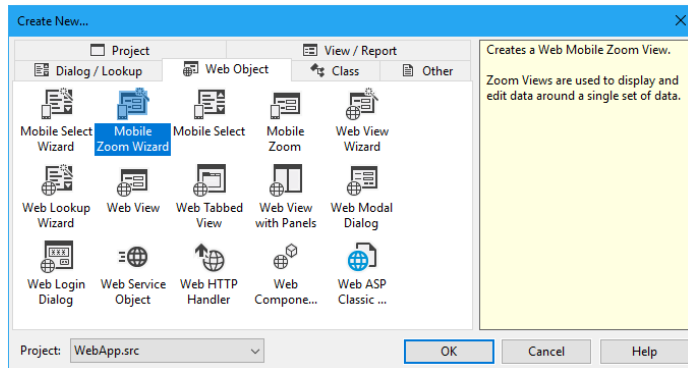
Module Types

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.

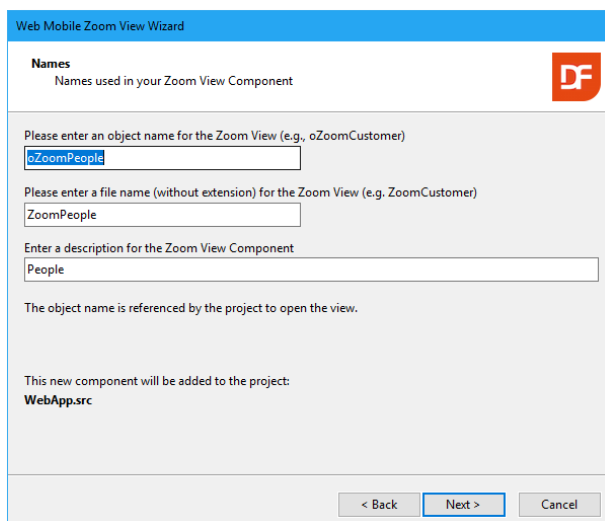
Following the suggested naming conventions in the wizard, enter the following information while processing the wizard:



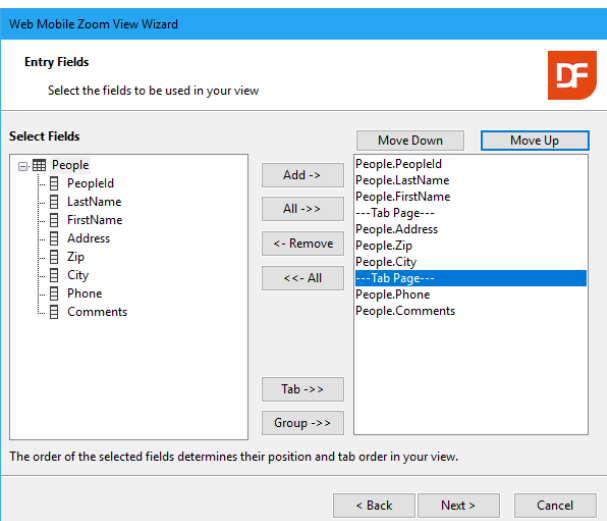
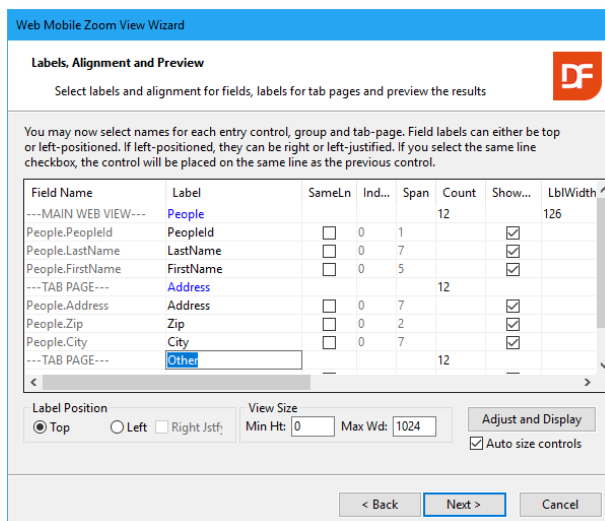
- ☒ Create a Simple Form Web Zoom View
- ☐ Create a Header/Detail Web Zoom View

- oZoomPeople for the object name
- ZoomPeople for the filename
- People for the description
- Choose 'Create a simple Form Web Zoom View'
- Choose the 'People – cPeopleDataDictionary' table

As shown, place all the columns on the View. Add two tab-pages as indicated so related items will be grouped together and there will be more space for entering comments.



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.

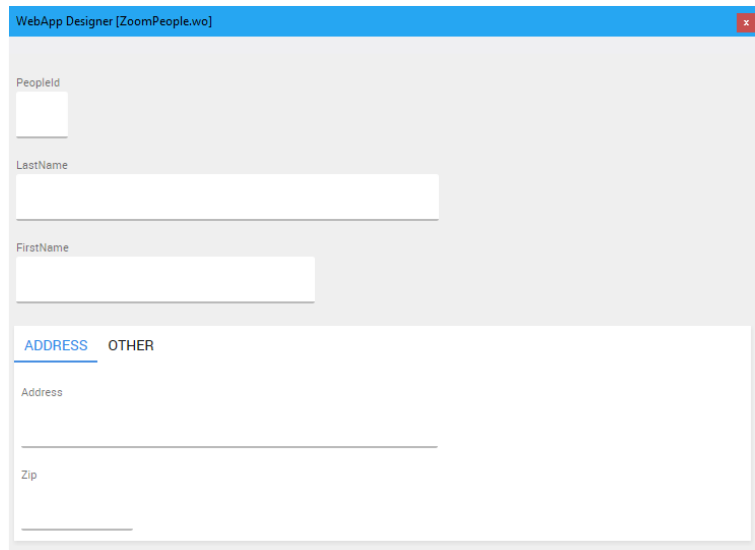
You can click the button labeled "Adjust and Display" to see what your web data entry view would look like. Web controls are laid out in a column based structure (this will be explained in the next chapter named "Layout and Positioning"). The number of columns (automatic or self-calculated) determine 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 to be taken

back to the Studio. In the Studio, the result of the wizard will be loaded automatically. You will see the source code for the web view we just created.

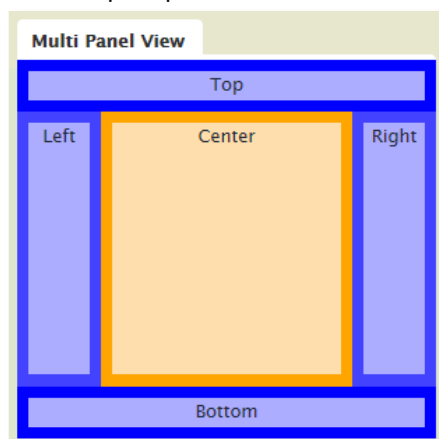
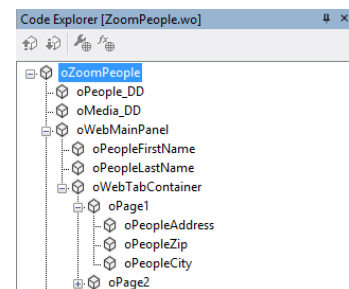
Press **F7** to view the layout in the WebApp Designer. By changing object properties you can change its layout, labels and more. The object properties are displayed in a panel that can be activated by pressing the **Ctrl+2** key-combination (or via the View menu-item, Properties option). If you did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



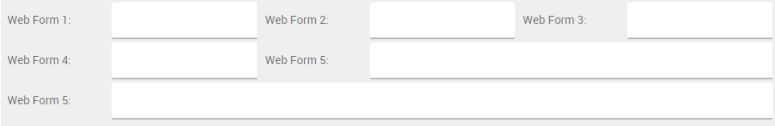
Layout and Positioning

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. To understand this better, 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

Column Layout View



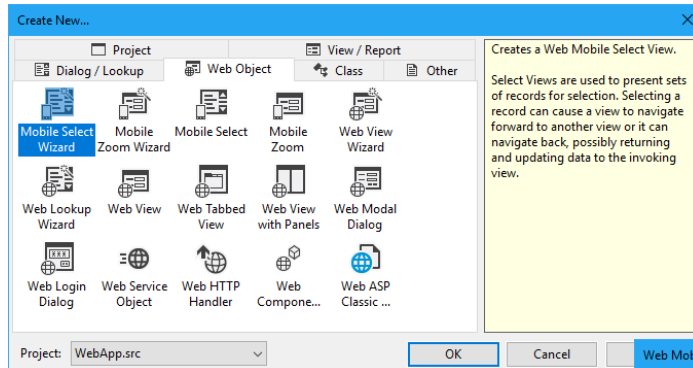
and it is the center panel.

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 on the same "line", they must share the available number of columns of the panel. 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 section, 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. Once again click on File, New, Web Object and select the 'Mobile Select Wizard' icon in the panel.

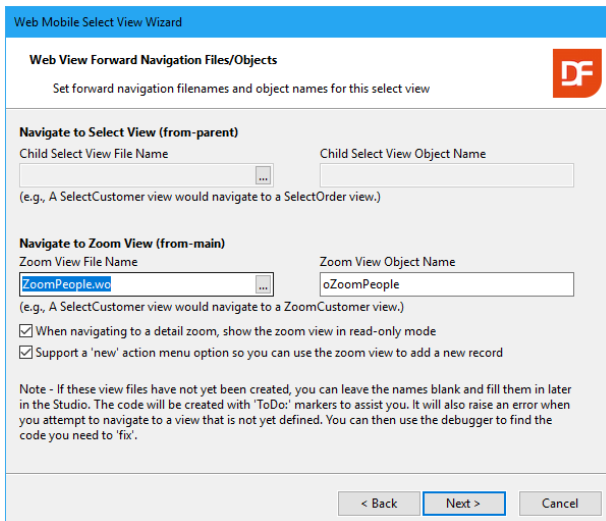
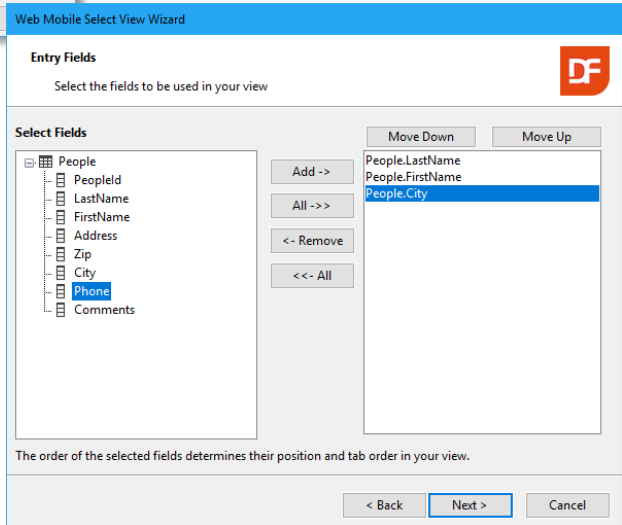
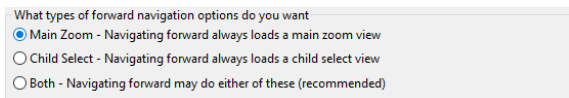


As with the Zoom View, follow the suggested naming conventions in the wizard, enter the following information while processing the wizard:

- oSelectPeople for the object name
- SelectPeople for the filename
- People for the description
- Choose the People table

In the next step, we can select the fields for the view. As this view will show a list of all the rows, we do not 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.



Now, we can determine where to the select view will take us to when zooming into the list. 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 grayed out detail in the row.

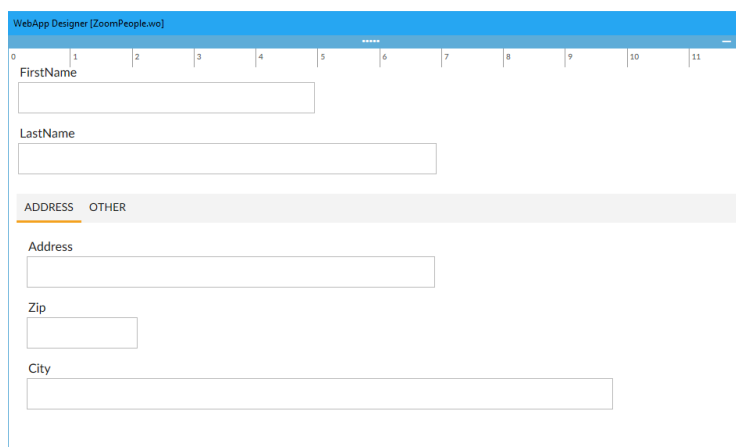
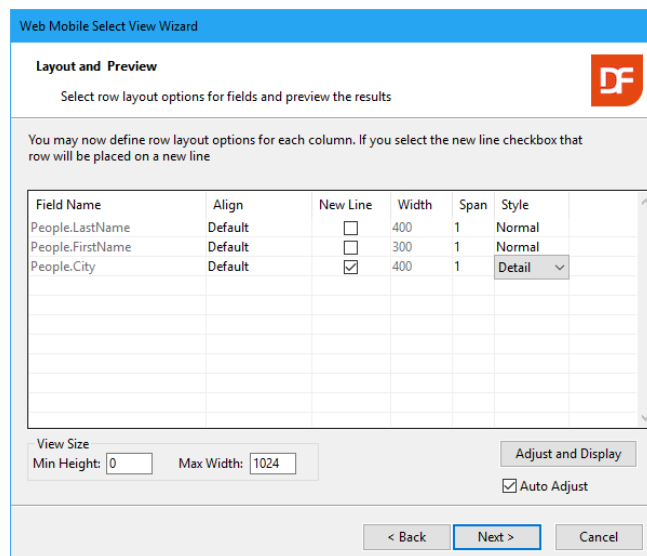
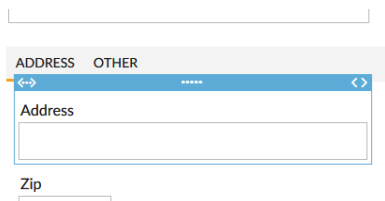
The New Line option increased the list row height, which makes touch navigation easier. Use the Adjust and Display button to preview the select view you are creating.

WebApp Designer Panel

After you have finished the wizard, your Select View will be shown in the Code Editor as well as in WebApp Designer. Press the **F7** key if the designer is not present. That key will display and hide the designer whenever pressed.

Select a 'control' via a mouse click in an input field or a container. The designer and the code explorer show the selected control. Click on a control, i.e. an input field or a container. The designer and code explorer show information on the selected control and 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 object, the designer shows an I-beam image 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

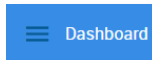
Testing

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, as soon as the compilation is successfully finished, the application starts by opening or attaching to your preferred (default) 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. We will tell you more about the login dialog soon.



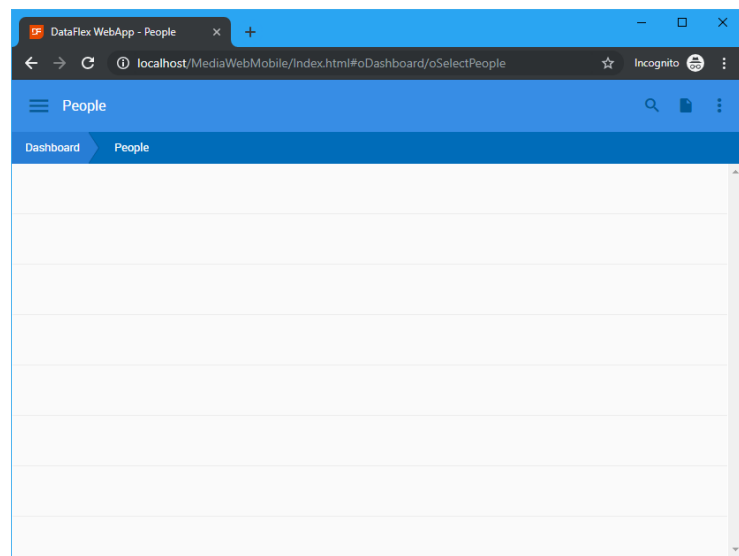
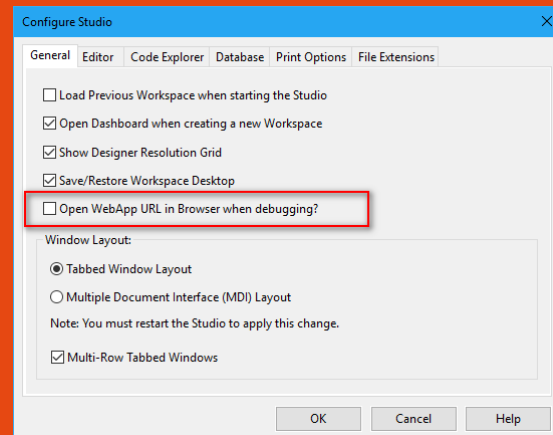
In the top left corner of the screen you will see three stacked bars. This icon represents the *hamburger menu*. This is the main menu of the application, where all navigation options will be added to. You can bring up the menu by clicking the hamburger icon.

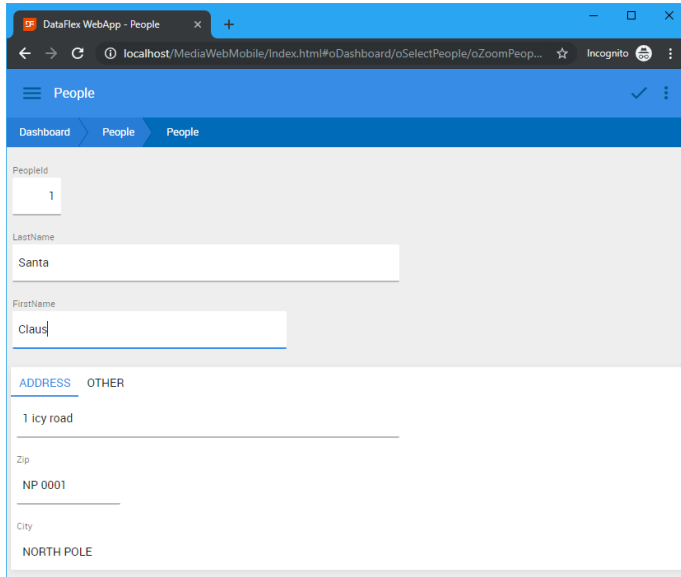
In this menu, we can select the People Select View by clicking on Views and then People. This will show a list of all rows in the People table.

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 click the save () button.

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 select 'Edit'.

Opening in read-only mode is the default, because 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:

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

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

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.

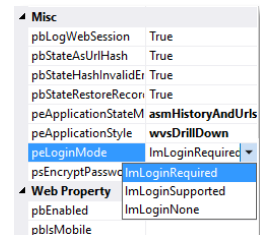
Close the browser application to return to the Studio. Sometimes closing the debugger does not close the run mode of the Studio. You can stop the running mode by clicking the stop button.



Login System

As you have seen in the preview topic, a login dialog is displayed as soon as the application starts, but that can be changed. 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 so no login dialog is needed. You might want to offer special features depending on the user accessing the application and, in that case, the use of the login dialog makes sense.

To make testing easier, you can turn off login during the development phase of web applications. To do so, make webapp.src the current tab-page in the DataFlex Studio and click on the oWebApp object in Code Explorer. In the properties panel locate the peLoginMode property and change it from lmLoginRequired to lmLoginNone. For applications that offer additional functionality after logging in, you may want to use lmLoginSupported instead.



Dashboard Tile Link to Login Page

The Dashboard, by default, shows the currently logged in user in the 1st 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:

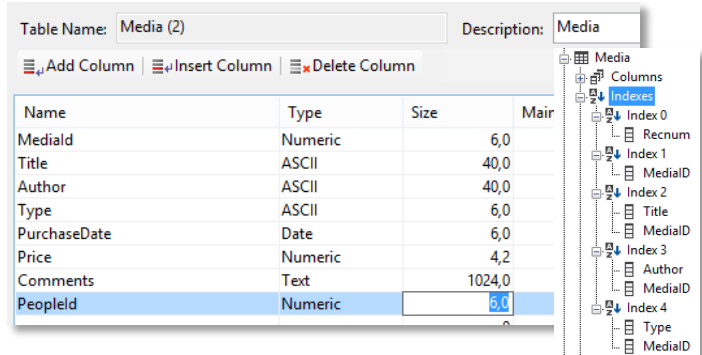
- Set the pbServerOnClick of the oWelcomeTile to true.
- Add the OnClick event via the events tab-page in the object properties.
- Write "Send RequestLogout of ghoWebSessionManager" in the OnClick procedure
- Insert data-ServerOnClick="LoginAgain" in the first div element of the HTML code of the UpdateHTML message in the OnLoad event. Without this the OnClick event will not fire!

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.
- The column PeopleID will be used 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.



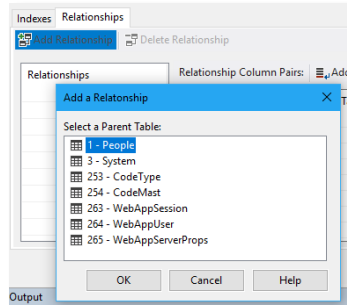
Name	Type	Size	Mair
MediaID	Numeric	6,0	
Title	ASCII	40,0	
Author	ASCII	40,0	
Type	ASCII	6,0	
PurchaseDate	Date	6,0	
Price	Numeric	4,2	
Comments	Text	1024,0	
PeopleID	Numeric	6,0	

MediaID will be the key field. Create an index for it as well as for Title, Author and Type. To make those indexes unique, add MediaID as last segment of each index and select 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 lists/reports. It is not difficult to create more indexes at a later time if you need this for certain lists/reports.

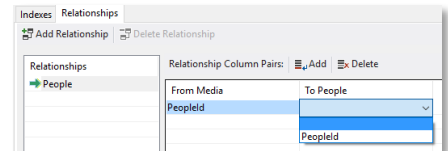
The Media table contains information about rows of our media in the possession of a certain person. In technical

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



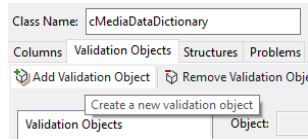
The selection of the parent table opens the option to specify from which child column(s) to which parent column(s) the relationship is made. The type and

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

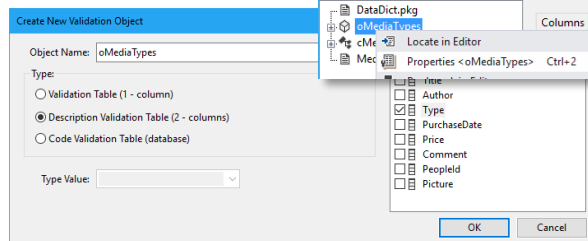


The Business Rules for the Media table

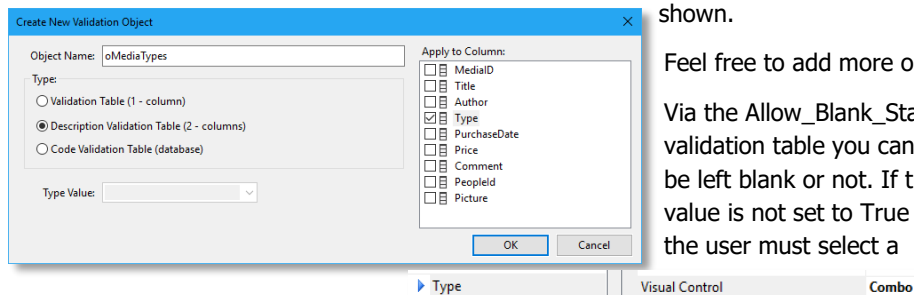
Finally, we will add some more Business Rules in the Data Dictionary. For this, the table needs to be saved first. The MediaID column will be a Key Field and the Title column is required. You should be able to do this with the guidelines given with the People data dictionary.



The values for Type should always be entered in uppercase (the Capslock attribute needs to be selected), but let's add something extra. We want the user to use consistent naming when entering Media Types. Enter 'CD' if it is a CD-Rom, enter 'BOOK' if it is a book. Not consistently entering such details makes report selections (such as 'Show me all books') quite difficult. Therefore, we will make a simple validation table on the column Type. You do this via the Validation Objects tab-page. Click the "Add Validation Object" button select the 'Type' column under 'Apply to Column' and 'Description Validation Table' for the type. Enter 'oMediaTypes' for the object name. Object names at this level need to have a unique name.

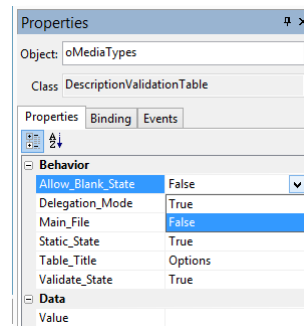


After you clicked the OK button you can start entering values for the table. We suggest you enter the values as shown.



Feel free to add more optional Types.

Via the Allow_Blank_State property of the oMediaTypes validation table you can indicate whether the value may be left blank or not. If the value is not set to True the user must select a



value from the list when creating or editing a record. Configure this setting as you please.

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: As a side note, open the tab-page called Structures and you will see that the People table 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 us 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 is 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 a data entry view in a less automated way and will have more control over what happens.

From the menu under File, choose New, Web Object, but now click on: Mobile Zoom.

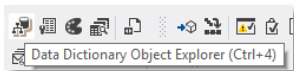
After that, enter "oZoomMedia" for the object name and the file on disk will be named ZoomMedia.wo.

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

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

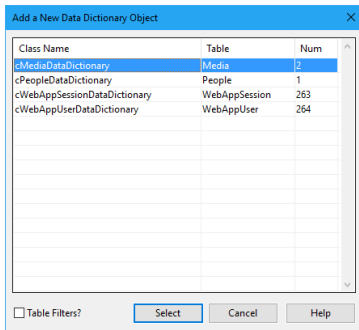
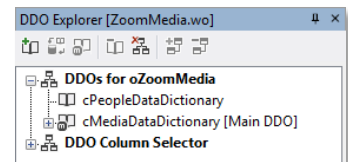


table Media, select the cMediaDataDictionary class (highlighted in the picture).

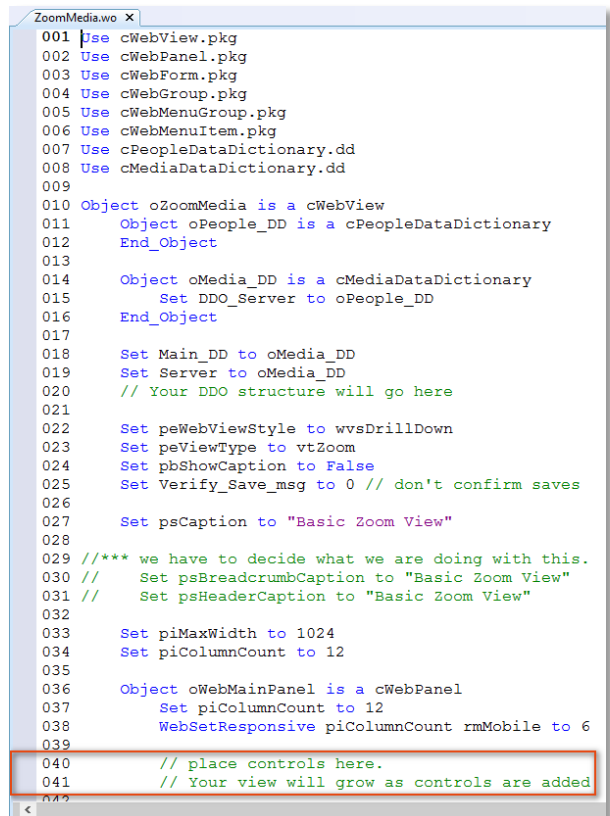


The Studio automatically makes the DDO for the Media table the main DDO and because Media has a relationship to a parent table (People), the DDO for that table will also be automatically added. When the choices are not correct, you can change these via the floating menu on each of the DDOs and apply the change. In our example this is now correct, so no action needed.

From the DDO Column Selector select all columns of Media (select all but the Recnum checkbox) 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.



```


001 Use cWebView.pkg
002 Use cWebPanel.pkg
003 Use cWebForm.pkg
004 Use cWebGroup.pkg
005 Use cWebMenuGroup.pkg
006 Use cWebMenuItem.pkg
007 Use cPeopleDataDictionary.dd
008 Use cMediaDataDictionary.dd
009
010 Object oZoomMedia is a cWebView
011 Object oPeople_DD is a cPeopleDataDictionary
012 End_Object
013
014 Object oMedia_DD is a cMediaDataDictionary
015 Set DDO_Server to oPeople_DD
016 End_Object
017
018 Set Main_DD to oMedia_DD
019 Set Server to oMedia_DD
020 // Your DDO structure will go here
021
022 Set peWebViewStyle to wvsDrillDown
023 Set peViewType to vtZoom
024 Set pbShowCaption to False
025 Set Verify_Save_msg to 0 // don't confirm saves
026
027 Set psCaption to "Basic Zoom View"
028
029 /*** we have to decide what we are doing with this.
030 // Set psBreadcrumbCaption to "Basic Zoom View"
031 // Set psHeaderCaption to "Basic Zoom View"
032
033 Set piMaxWidth to 1024
034 Set piColumnCount to 12
035
036 Object cWebMainPanel is a cWebPanel
037 Set piColumnCount to 12
038 WebSetResponsive piColumnCount rmMobile to 6
039
040 // place controls here.
041 // Your view will grow as controls are added

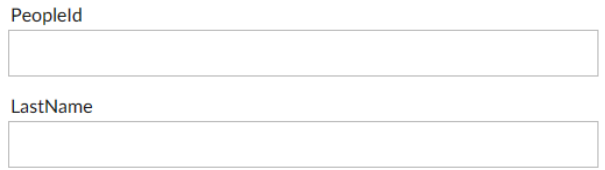
```

Show LastName & FirstName Together

Would it not be better if the People LastName input field was positioned after the PeopleId input field, instead of underneath it? Yes, of course. Since we covered how the layout system works, you should be able to make that change. Each container is divided into columns (also indicated in the data entry wizard). The default number of columns for a container is 12. An input like the PeopleId does not need 12 columns or the full width of the cWebView and that "line" can be shared with other controls, such as our "LastName" control.

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

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



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

After changing the piColumnSpan, the LastName object does not move yet as its piColumnIndex (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.

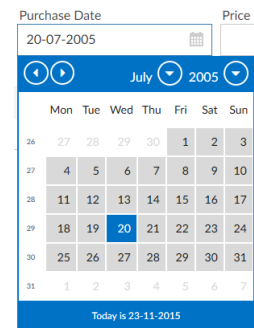


Date Selector

The input controls created by the wizard or the templates are "simple" controls. With a minimum of effort, you can make controls more powerful – e.g. present an icon that a date may be selected from a date selector box, which gets displayed as soon as the user clicks the icon.

Locate in the source code the oMedia_PurchaseDate object and change the class name 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 corresponding DataFlex class necessary to make the control work.

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 class name 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 instructed to grab the image stored on 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 of AppSrc, Data etc. To retrieve the folder location we will create a method (a function) in the oApplication object (found in the WebApp.src file). Change the oApplication object as follows:

```
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 there. Use – for now – Database Explorer (or the Table Viewer) to connect a Media row to an image by placing in the name of the image file in the column Picture.

To show the image stored, 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:

Title			
20 Great Love Songs			
Author			
The Beach Boys			
Type	Purchase Date	Price	
CD-Rom	20-07-2005	23.75	
PeopleId	LastName		
1	Claus		
Comments			
Twenty nice love songs			
			

```

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", then press Ctrl+X to cut the code, move the insertion cursor in the code to the new location and press Ctrl+V
- An easier way would be 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 to the end. This object order change can also be done via the buttons in the tool-bar of the Code Explorer or by selecting the corresponding menu options in Code Explorer's floating menu
- The WebApp Designer offers the third way to move the object: select the object and drag it to the location behind the LastName object

Finally, to give more space to Comments, locate the property pbFillHeight of the oMedia_Comments object and set it 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 its use.

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 will not know the corresponding ID 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 code:

```
WebRegisterPath ntNavigateForward oSelectPeople

Procedure OnPrompt
    Send NavigatePath
End_Procedure
```

This will make a lookup icon appear in the PeopleID form and if clicked, users navigate to the SelectPeople view.

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 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 data: a select view and a zoom view. We have to make some code changes to zoom in on a Media row 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.

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 that in 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 will be able to navigate from the select-view to the zoom-view when you click on 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. The reason is that this component was 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 should provide a 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. To do that, start by opening the Dashboard view and locate the object named oTile2 and 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 desire, 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 3rd tile to the Select Media view based on above information.

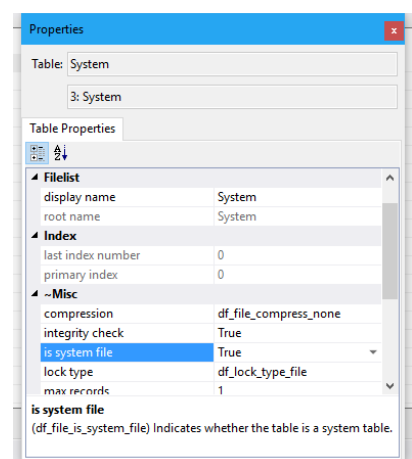
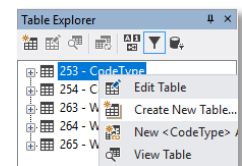
Automatically Generate Key Fields

For People as well as Media we defined unique, numeric keys. This number stored in those key fields is not relevant to the user and it would be even troublesome for users to try to remember what the last given number was when they try to create a new Media or Person row. In order to make the interface work better with those premises in mind, we will add a system table to the application:

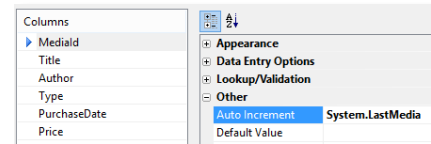
1. Create an extra table. In this table we will always store only one (1) record. This is called a system table. In this table we define two columns only: LastPeople and LastMedia. These columns are numeric, 6 digits.
2. The Data Dictionaries of Media and People will take care of generating these ID's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

Table Name: System (3)		Description:	
Add Column Insert Column Delete Column			
Name	Type	Size	Ma
LastPeople	Numeric	6,0	
LastMedia	Numeric	6,0	

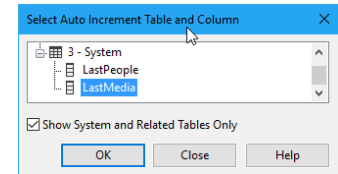
To make sure it actually becomes a system-file, select True for the "is system file" table attribute. Save the table structure before continuing.



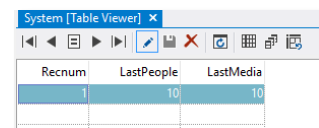
We want the Data Dictionaries to automatically increment the IDs each time a new row is saved. Open the data dictionaries for the tables People and Media in the Studio. Look for the attribute Auto Increment (grouped under Other) in the list of properties for the columns PeopleID (in People) and MediaID (in Media) and click the prompt button. From the list of tables, select System and from the columns, the correct source data column – those are LastPeople for PeopleID and LastMedia for MediaID.



Note: The value can be taken from a system table or a parent table. That is why you see the checkbox labeled "Show System and Related Tables Only".



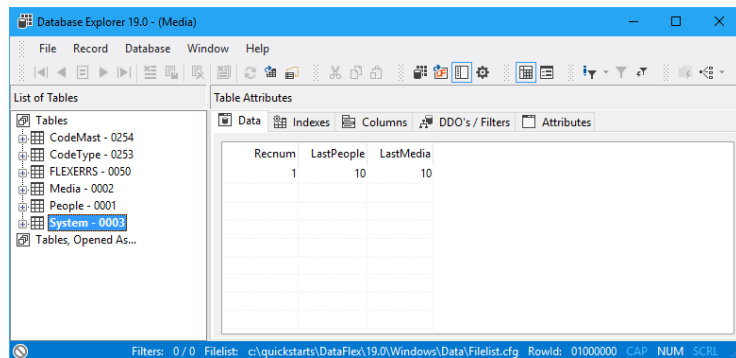
Ok. This should work, if not it is because you have already created some records, with ID's 1, 2, 3 etc. The first time we will use our automated increment function it will try save a record with ID of '1' and that will not work because that value already exists. ID's should always be unique – it is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

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 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 by selecting File | Configure and look for "Open/Set Tables Readonly" under Flags and Table tabs and unselect it. Then press OK to save your setting.



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 appear as tool-tips 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. Typed values will always be uppercase, a selection can be made from 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, 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 would it not be great if we could browse through People and display all media that each person has borrowed? 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 opened. Clicking on 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 by right clicking and selecting delete from the floating menu.



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. Locate the cWebColumnImage entry on the Class Palette and drag it onto 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.

All Media		
1492: The Conquest Of Paradise	Vangelis	
20 Great Love Songs Twenty nice love songs	The Beach Boys	
20 Greatest Hits Very nice song	Jackie Wilson	
24 Great Jazz Performances (Disc 1)	Glenn Miller	

Wildcard Search View

Would it not 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 control to the view. 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 and divide the rest of the default number of columns (= 12) between the checkbox and the button
- Label the form "Filter on:"
- Label the checkbox "Case Sensitive"
- Label 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 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
```

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

The properties will get their value when the search button is clicked. 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

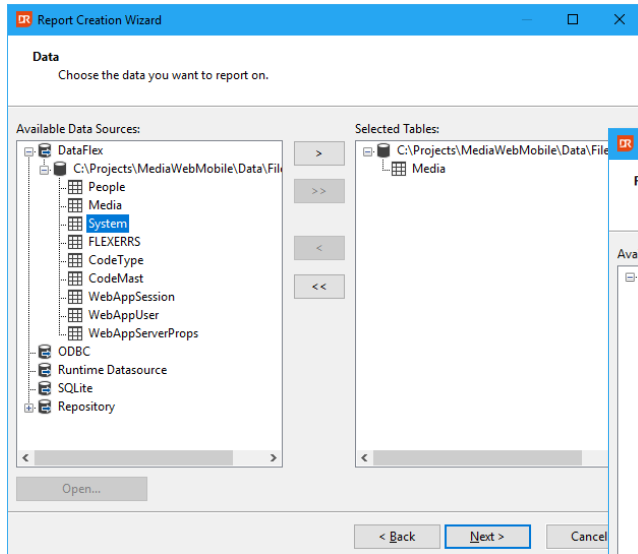
Reports and Lists

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. The report integration 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 adjust the sort order,

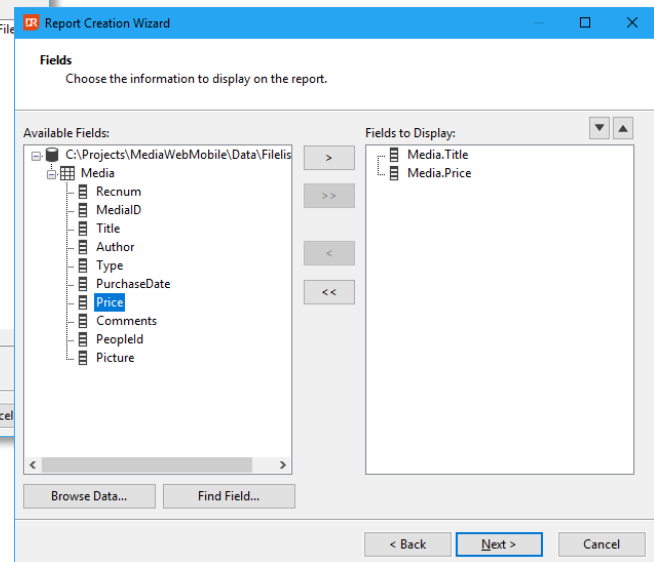
output device, as well as the selection criteria. It is simple: First create a report with DataFlex Reports, and then start the wizard – the rest is self-explanatory.

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 repeats for each row that matches selection criteria set for the report. In the 'Fields' page, select the columns Title and Price.

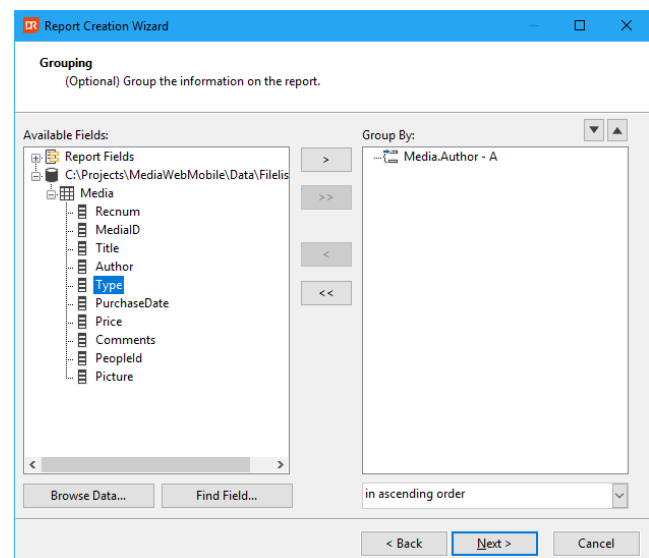
The next wizard page let you select the column(s) to group data on. Here we select the column Author. This means that printing of titles per author is possible.

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

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

Look at the screenshot of the report in the designer and see how we used colors to 'beautify' the report. We also added a function to combine the Author name with the number of Media rows we have for this author. Each Author name has a total on Price of their titles and the page footer contains a 'Page N of M' text.

A feature that is not shown but interesting: we can sort the details of each group on Title or any other column.

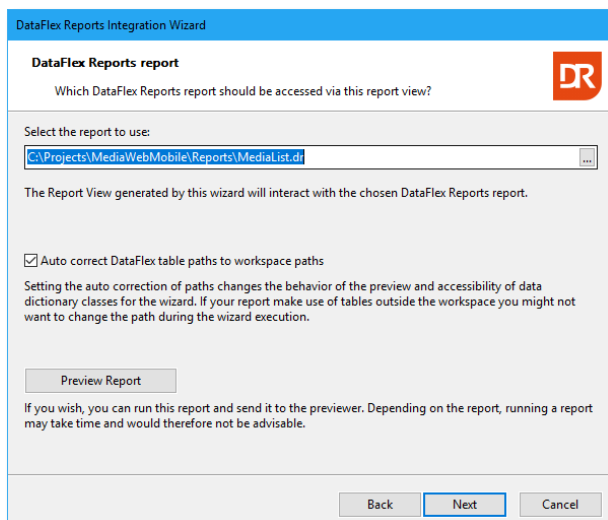


DataFlex Reports developer edition optionally installs an integration library. Attach the workspace to the library by selecting Tools, Maintain Libraries. In the dialog that pops up you will see the already connected libraries.

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 called Libraries inside the DataFlex environment. If not look for \Libraries in the root of the installation disk (e.g. c:\libraries). You select the location during the installation of DataFlex Reports Developer Edition. After clicking OK, a wizard starts that guides you through the process of attaching the library. 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.

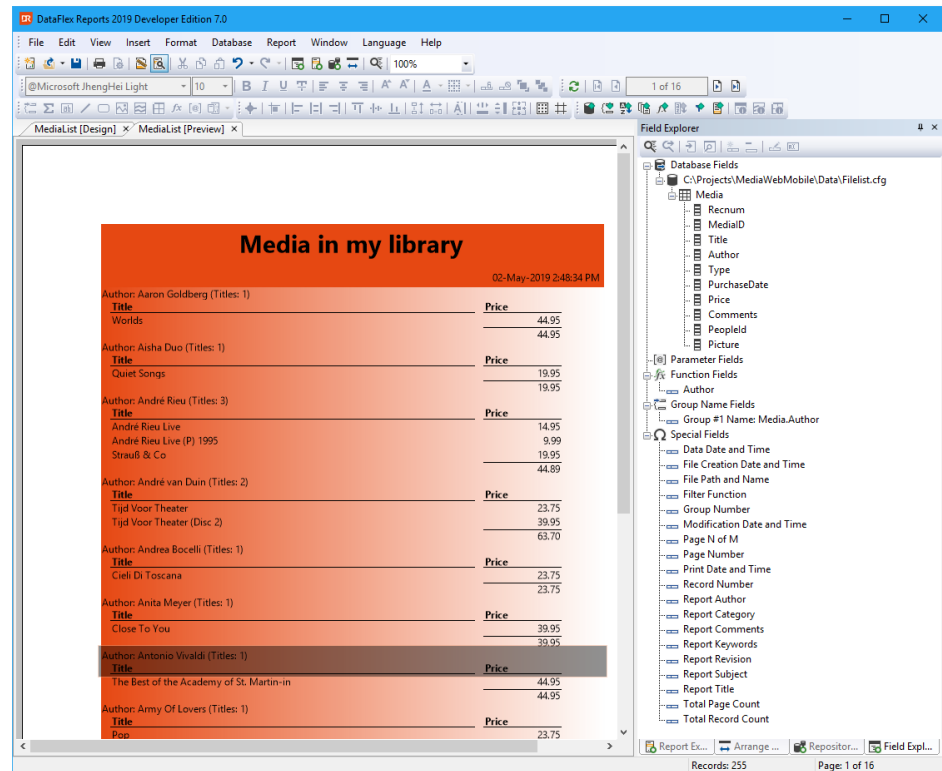
In the wizard you can choose between connecting to an existing report or create a new RDS based report. Choose connect to an existing report. Then, on the next page, select the report that you want to integrate. You can preview the report but keep in mind that a preview might take some time when you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

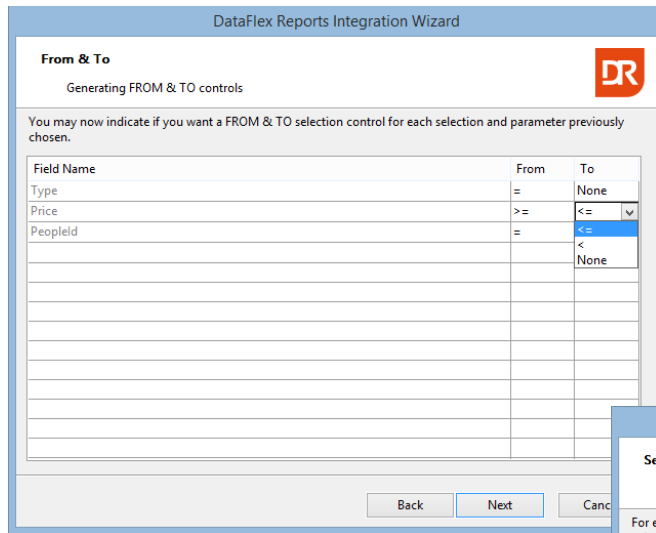


After clicking the 'Next' button you can select / confirm the application style. Use the 'DrillDown – Mobile/Touch' option.

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 when you do not want to build a 500+ pages report – it takes too long for the average impatient web user and it makes no sense to view so many pages over the web. You might even want to check if the user made selections or not.





From & To
Generating FROM & TO controls

You may now indicate if you want a FROM & TO selection control for each selection and parameter previously chosen.

Field Name	From	To
Type	=	None
Price	>=	<=
PeopleId	=	<
		None

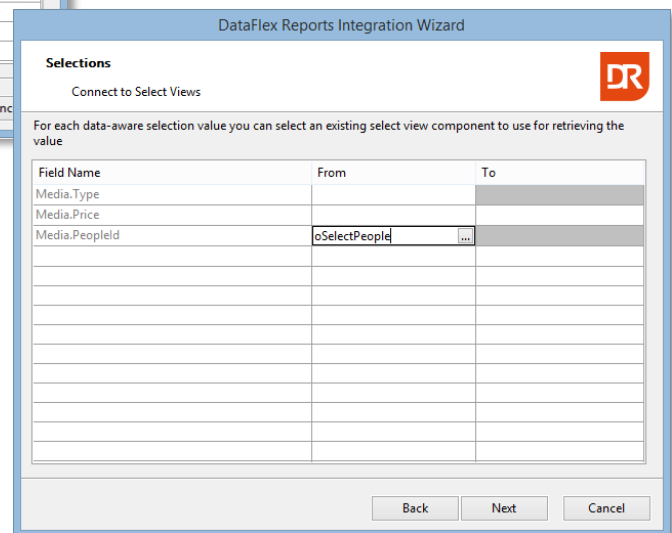
Back Next Cancel

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

Also based on selection criteria, you may specify the filter operator and whether you want a 'from-to' selection or selection on just one value.

Change the operator for the Price column to 'greater than or equals' for the 'From' column and 'less than or equals' in the 'To' column.

In the next wizard page you can connect a selection control to a web selection view. The PeopleId column can be connected to the select view for People made earlier in this Quickstart. To do this, place the cursor in the 'From' grid item for Media.PeopleId and click the prompt button. Then select the 'SelectPeople.wo' file from the Windows common file dialog. On *return* the object name of the select view will be shown in the grid.

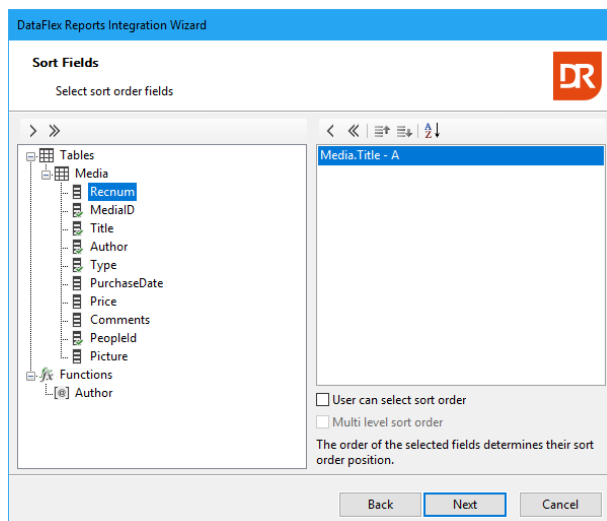


Selections
Connect to Select Views

For each data-aware selection value you can select an existing select view component to use for retrieving the value

Field Name	From	To
Media.Type		
Media.Price		
Media.PeopleId	oSelectPeople	

Back Next Cancel



Sort Fields
Select sort order fields

Tables

- Media
 - Recnum
 - MediaID
 - Title
 - Author
 - Type
 - PurchaseDate
 - Price
 - Comments
 - PeopleId
 - Picture
- Functions
 - Author

Media.Title - A

☐ User can select sort order

☐ Multi level sort order

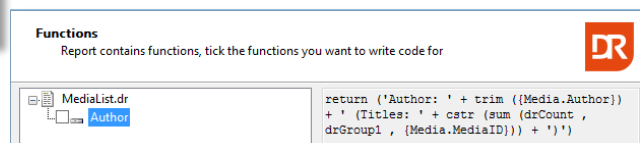
The order of the selected fields determines their sort order position.

Back Next Cancel

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

If the report contains formulas you will see a wizard page showing the formulas and you can select whether you want the wizard to generate code to change the

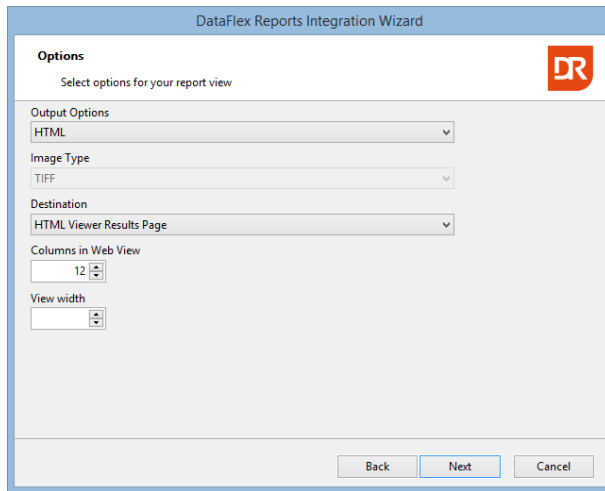
content of a formula. The MediaList report contains one formula but it is not a candidate to be changed at runtime.



Functions
Report contains functions, tick the functions you want to write code for

Function	Code
MediaList.dr	return ('Author: ' + trim ([Media.Author]) + ' (Titles: ' + cstr (sum (drCount , drGroup1 , (Media.MediaID))) + '))'

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



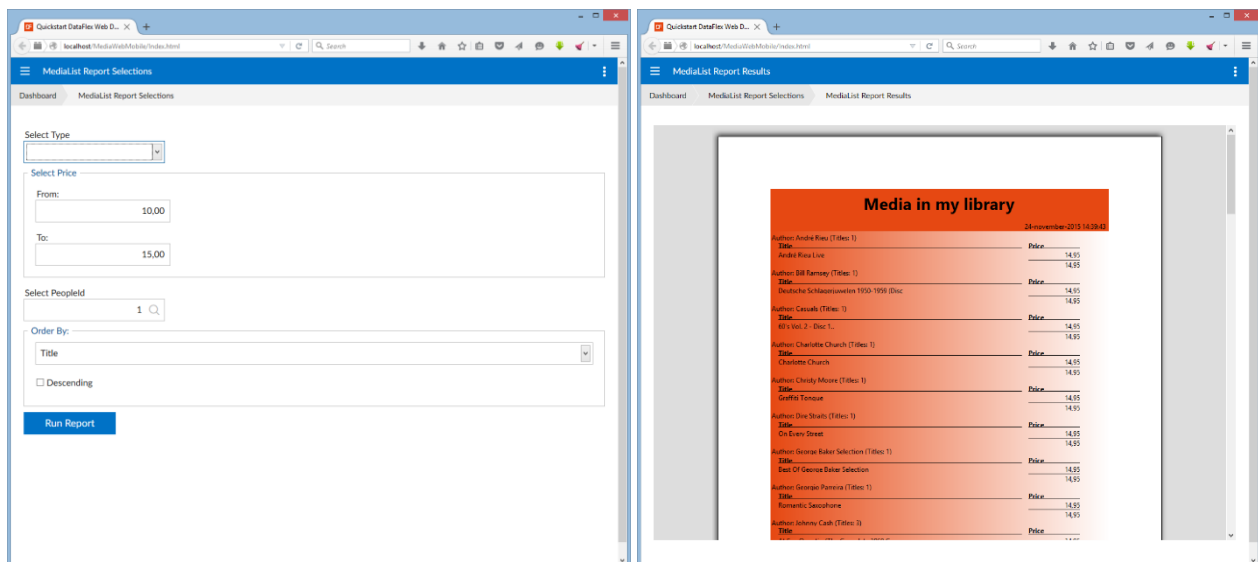
For (DrillDown Mobile-Touch style) web applications, the default will be HTML through a HTML previewer control. If the user wants an extra/different output, they can select 'Export' after viewing the results.

The wizard will create one or two web components based on the choices made during this wizard session. In our case two components will be created and you have to specify (or simply accept) the names for the objects and component filenames twice. As suggested names, the report name is appended with 'Select' and 'Results'.

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 their 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

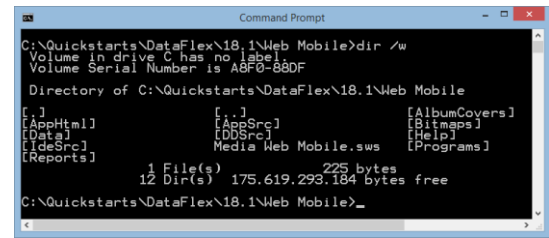
Go to the Create New dialog in the Studio one more time and create a Select View. Do not use the wizard as the data will not be read from a database but from the file system. Use oSelectAlbumCover as the name of the select view. Select the cWebList object and change the name to oFilesList. Change the 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, so 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 means it is a folder. This is not a DataFlex invention, it is how the operating 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 from that unique identifier. It also is needed to identify for the caller object what file the user selected. The second use is to show the name of the file to the user and the third 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
        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 plus 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; we will 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. However, 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. In that case, OnShow is a better option. So add:

```

Procedure OnShow
    Send GridRefresh of oFilesList
End_Procedure

```

The event only fires if we 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 therefor 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 navigates to the select album cover image but sends a message to the cWebImage object instead. 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.

```

WebRegisterPath ntNavigateForward oSelectAlbumCover

Procedure SelectImage
    Boolean bEnabled

    WebGet pbEnabled to bEnabled
    If (bEnabled) Begin
        Send NavigatePath
    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 above 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 being 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. You need to register the upload folder in your web application as a valid upload folder. 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
```

After that, users may press the upload button when navigating to this view to select files from the local device and upload them to the server . Add code to check the MIME type to avoid uploading a non-image file.

Lastly, we alter the 'New' button in the oSelectAlbumCover object's ActionGroup so that selecting this option takes 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"

WebRegisterPath ntNavigateForward oUploadAlbumCover

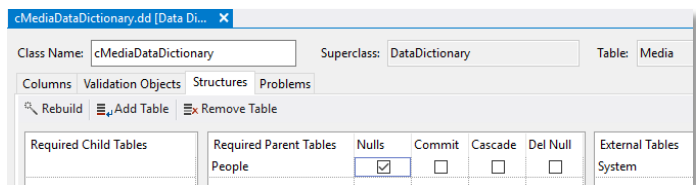
Procedure OnClick
    Send NavigatePath
End_Procedure
End_Object

```

Borrowed or Owned?

Relationships between tables, such as the Media and People tables, by default require an existing parent row. In this Quickstart it means that each Media row is connected to a row in the People table. If we see the Media as our collection of books, CDs etc and see the connection to People really as "lent to", we can make the relationship optional.

To do this, open the Media data dictionary and open the 'Structures' tab-page. The second list shows the required parent tables. Tables listed there are validated on each save. By selecting the 'Nulls' checkbox, we make the relationship to the People table optional.



Do not select the 'Commit' checkbox as it would not allow us to change the connection to a person anymore, and thus, defeats the idea of lending collection items.

Remove and Borrowed

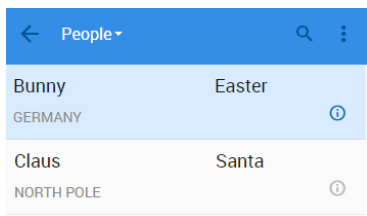
If you want to allow the application to delete a row from the People table, the relationship between the tables Media and People require you to make a decision about what to do with the Media that links to the person to be removed from the system. By default, the framework will delete all rows from the Media table connected to the person to be removed. You could decide to write off the collection item, but why? It would be better not allow removing the person when they have borrowed collection items. To do this, look up the option `Cascade_Delete_State` in the People data dictionary class and select False (default is True).

Alternatively, you can let the system remove the link between the Media and People row when removing the person. This means that you once owned the collection item but that it cannot be found anymore in your collection. If you want to allow this, keep the `Cascade_Delete_State` as is but check the option 'Del Null' in the Media data dictionary.

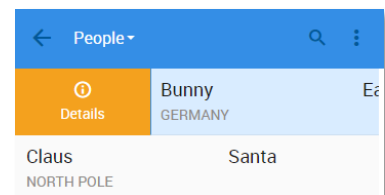
Swipe Buttons

Mobile devices offer swiping and this makes it possible to enhance to the application when used on a mobile device such as a smartphone.

A `cWebList` object usually contains two navigation options: one to drill-down to the next level (e.g. from customer to orders) and another to view the details of the current row. The width available for a list on a mobile device is limited, thus accommodating a pretty wide details button to go to details of a row is a challenge.



As an alternative, a swipe button, a class that hides the button until you swipe the list item horizontally, can replace the details button.



You can even have both the details and swipe buttons, hiding them based on the current device.

To implement the swipe button change the class name of the oDetailsButton to cWebListSwipeButton and change a couple of properties. Replace the current property settings with:

```
Set psCaption to "Details"
Set piWidth to 100
Set pbPositionLeft to True
Set psCSSClass to "WebIcon_Info Highlight"
```

If you want to have a details button and a swipe button, copy the button before replacing the properties and implement the OnNavigateForward event in the view, adding the following code to the event:

```
Integer eMode

WebGet peMode of ghoWebApp to eMode
WebSet pbRender of oDetailsButton to (eMode = rmDesktop or eMode = rmMobileLandscape ;
    or eMode = rmTabletLandscape)
WebSet pbRender of oSwipeDetailsButton to (eMode = rmTabletPortrait or ;
    eMode = rmMobilePortrait)
```

Peek Look at Borrowed Media

A couple of sections back (see Show All Media Borrowed) we showed how to drill-down from People into Media where you see the Media borrowed by a person. Let us make the application a bit more advanced and show the first four borrowed Media rows when clicking on a row in the People Select view instead of always navigate to select Media.

Open the Class Palette and locate the cWebListExpandPanel under Web Containers and drag this to the cWebList object in the SelectPeople.wo component. Name the object oMediaBorrowedExpandPanel and set peMode to wleManual.

Move the oDetailButton object from the cWebList object into the cWebListExpandPanel object. Then, since oDetailButton is not a child of the cWebList object any longer, change its class from cWebListColumnButton to cWebButton. Change the psCSSClass of the button to "AsAnchor" and set peAlign to alignLeft. Set the psCaption to "Person details".

Add another cWebButton object to the cWebListExpandPanel container and name it oViewMoreButton. To show the button on the right hand side of the container, set the peAlign property to alignRight. Set the psCaption to "View More" and set the psCSSClass to the same as the oDetailButton, "AsAnchor".

Clicking on the oViewMoreButton button should navigate to the oSelectMedia. For that to happen, add the following code to the button:

```
WebRegisterPath ntNavigateForward oSelectMedia

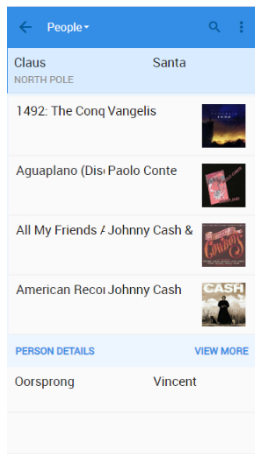
Procedure OnClick
    Send NavigatePath
End_Procedure
```

The next step is to add a DDO for the Media table to the select view. Since Media relates to People the Studio adds the relational constraint filter line automatically. Make sure to keep that line. Set the Ordering property of the DDO to 5, which means that the requests for data always use the index made from PeopleId, Title and MediaId.

Now open SelectMedia.wo and copy the cWebList object code and paste it into the cWebListExpandPanel object. Change the data binding of the cWebList object to oMedia_DD, adjusting the cWebList to read data from the correct table.

The copied cWebList has its psCSSClass set to "MobileList", which is fine, but adding "SubList" to it make the sub-list look better, so change the psCSSClass to:

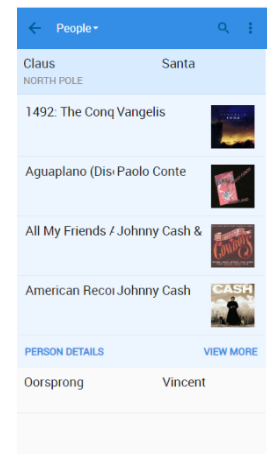
Set psCSSClass to "MobileList SubList"



The images on the left and right shows how the user interface without "SubList" (left) and with "SubList" (right) look like. Notice the difference in the light blue background color between the two.

To only show a couple of Media borrowed, set the piLimitRows property to 4 and set pbScroll to False to limit the height of the cWebList object for borrowed Media to the number of rows in the list. Set the pbShowSelected property to False to hide a current row marker in this sub-list.

When the person selected (name clicked on) has not borrowed any media, the list should display "No Media Borrowed". To make this happen, set the psPlaceholder property to this text.



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:

- <https://www.dataaccess.com/>
- <https://learning.dataaccess.com>

Other related sites:

- <https://support.dataaccess.com/forums>
- <https://www.dataaccess.com/dynamica> (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!



Discovering DataFlex

FOR WINDOWS

Learn how to build powerful Windows applications with DataFlex
By Vincent Comproing