ALTOVA®

How To Guide

Creating Elegant User Interfaces for Cross-Platform Mobile Apps

Contents

Introduction

Smartphones and tablets are everywhere, and superior mobile apps create a population of demanding users with high standards for performance. This can be challenge for enterprises that want to build mobile apps to support internal business goals.

Complicating mobile app development, many companies have adopted BYOD (bring your own device) policies to give end users choice and flexibility while off-loading mobile device procurement issues. However, BYOD scenarios present significant hurdles for mobile app developers, who now must build apps for multiple mobile operating systems, OS versions, and screen sizes.

In the consumer market, we see desktop and even notebook computers becoming less popular as more smartphone and tablet users depend on mobile devices for all their connectivity needs. In contrast, mobile apps must frequently bridge computing environments. A busy executive who wants to check enterprise data on his phone over breakfast will then commute to the office and demand the same information in the same format on a desktop computer. Corporate sales people will want their relevant enterprise apps on both their laptops and mobile phones. Both desktop and mobile applications need up to the minute access to well-established backend data systems.

Enterprise developers in this environment need a tool that lets them quickly deliver a new mobile app or an updated version. MobileTogether lets developers design, test, and release one version of an app to run on all mobile devices, with interface features that will delight users across platforms.

Combining rapid mobile app development (RMAD) with a single development environment for building cross-platform apps makes Altova MobileTogether an ideal tool to create specialized mobile apps for enterprise operations and other data-centric operations.

The development schedule using MobileTogether can be as short as a few days vs. weeks or months for traditional native platform coding. That kind of productivity makes it practical to build specialized apps that might only be installed by a small number of users, but greatly improve efficiency.

Using MobileTogether means you no longer need to outsource your mobile app development or hire a large team of very expensive mobile developers (essentially one per mobile OS platform).

With these enterprise goals in mind, let's look at some real-world apps to see how we can leverage the features of MobileTogether to create elegant and intuitive user interfaces – across all mobile operating systems, screen sizes, and devices.

About MobileTogether

Altova MobileTogether is an affordable framework for building data-centric apps for all platforms. This guide highlights MobileTogether user interface design features with tips and tricks to help developers create elegant applications for all mobile devices.

For readers who may not be familiar with the MobileTogether development framework, the remainder of this section describes MobileTogether benefits and provides a brief description of the steps followed to create and deploy a mobile app with MobileTogether.

MobileTogether Advantages

For app developers:

- MobileTogether reduces the time required to build apps for all mobile platforms from months per platform down to a few days with **automatic code-generation** for all mobile platforms from one design environment.
- The MobileTogether framework **includes the backend server** for the mobile apps, so when you design and build your mobile app with MobileTogether you're developing the front-end app and the back-end server logic at the same time and in one environment!
- MobileTogether **generates native apps** for all supported platforms from a single design. MobileTogether is a full-fledged mobile development environment that includes powerful data integration capabilities, graphing and charting, and sophisticated control-flow and business logic processing, and access to mobile device features like GPS, camera, email, text messaging, and more.
- MobileTogether features affordable pricing. The MobileTogether Designer is free to download and use to create and test any number of mobile apps. Pricing for the backend server is based only on server performance (i.e. number of CPU cores), not on the number of apps you wish to host or the number of end users.

For business and enterprise customers:

- Apps can be built by an in-house IT or development team. There's no requirement for specialized mobile developers.
- Native apps are completed and delivered to end users on their mobile device of choice – smart phone, tablet, laptop, or all three – in record time.
- MobileTogether makes it simple to build sophisticated apps that connect to existing backend systems, mobilizing critical data to increase end user productivity and satisfaction, while delivering tangible ROI.

Supported Platforms:

- Android
- iOS
- Windows 8, Windows 10
- Windows Phone 8
- HTML-5 Browser Based Client

Steps to Develop and Deploy a Mobile App for All Platforms:

1. Developer or technical user uses the **MobileTogether Designer** to build a mobile app, with execution and debugging functionality in the built-in mobile device Simulator.

2. Developer deploys the app to your **MobileTogether Server** (either on-premises or in the cloud).

3. End users access the app – with robust security considerations – by connecting to your MobileTogether Server on their mobile device. They simply download the free, **MobileTogether App** from their app store of choice and connect to your server.

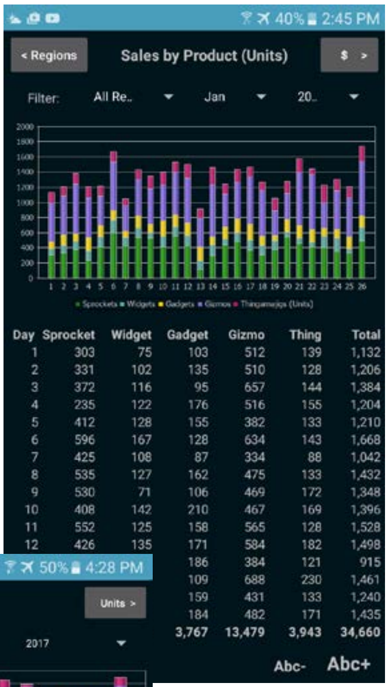
4. If further customization is required, developers add custom branding, **compile an app for each platform**, and submit it to the app stores.

The Company Sales Report Example App

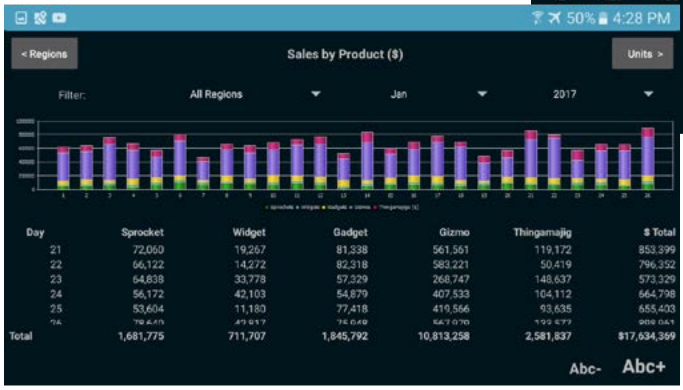
This section describes the Company Sales Report app referred to in the specific examples in the next section.

The Company Sales Report is a single page design created in the MobileTogether Designer for a cross-platform mobile app that presents information from a fictional company sales database. The company is organized into four sales regions and has five products for sale.
The app presents literally 44 unique views into the company sales statistics, cascading from two main Product and Region views, each reporting units or dollars, then drilling down to offer each individual product and region combination.

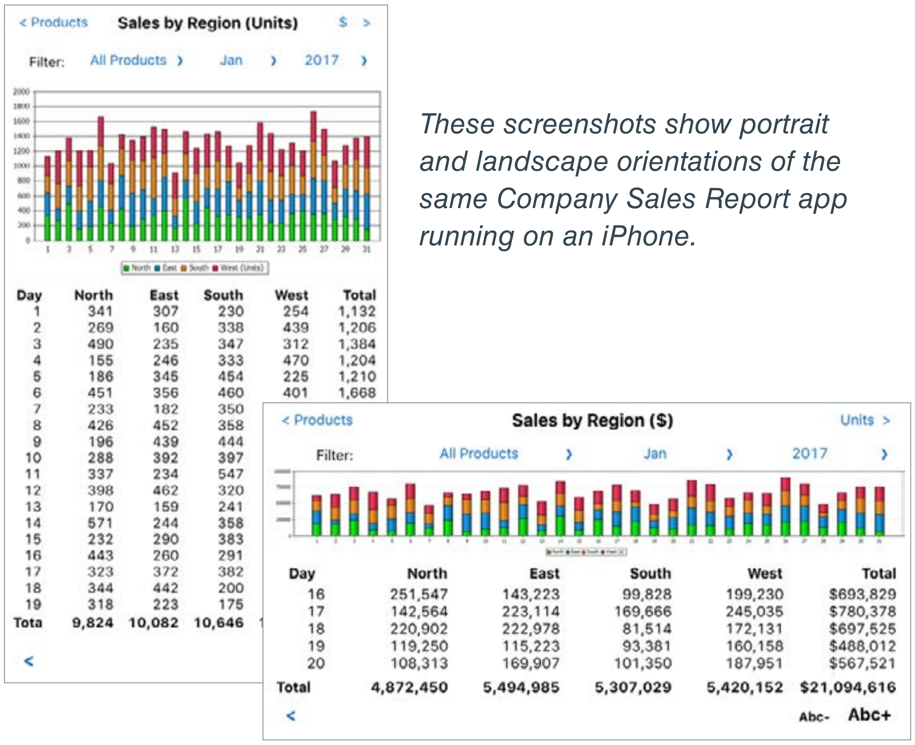
These screenshots show portrait and landscape orientations of the Company Sales Report app running on an Android phone.



Day	Sprocket	Widget	Gadget	Gizmo	Thing	Total
1	303	75	103	512	139	1,132
2	331	102	135	510	128	1,206
3	372	116	95	657	144	1,384
4	235	122	176	516	155	1,204
5	412	128	155	382	133	1,210
6	596	167	128	634	143	1,668
7	425	108	87	334	88	1,042
8	535	127	162	475	133	1,432
9	530	71	106	469	172	1,348
10	408	142	210	467	169	1,396
11	552	125	158	565	128	1,528
12	426	135	171	584	182	1,498
			186	384	121	915
			109	688	230	1,461
			159	431	133	1,240
			184	482	171	1,435
			3,767	13,479	3,943	34,660



Day	Sprocket	Widget	Gadget	Gizmo	Thingamajig	$ Total
21	72,060	19,267	81,338	561,561	119,172	853,399
22	66,122	14,272	82,318	583,221	50,419	796,352
23	64,838	33,778	57,329	268,747	148,637	573,329
24	56,172	42,103	54,879	407,533	104,112	664,798
25	53,604	11,180	77,418	419,566	93,635	655,403
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Total	1,681,775	711,707	1,845,792	10,813,258	2,581,837	$17,634,369


< Products
Sales by Region (Units)
$ >
Filter: All Products › Jan › 2017 ›
North East South West (Units)
<

Day	North	East	South	West	Total
1	341	307	230	254	1,132
2	269	160	338	439	1,206
3	490	235	347	312	1,384
4	155	246	333	470	1,204
5	186	345	454	225	1,210
6	451	356	460	401	1,668
7	233	182	350		
8	426	452	358		
9	196	439	444		
10	288	392	397		
11	337	234	547		
12	398	462	320		
13	170	159	241		
14	571	244	358		
15	232	290	383		
16	443	260	291		
17	323	372	382		
18	344	442	200		
19	318	223	175		
Tota	9,824	10,082	10,646		

< Products
Sales by Region ($)
Units >
Filter: All Products › Jan › 2017 ›

Day	North	East	South	West	Total
16	251,547	143,223	99,828	199,230	$693,829
17	142,564	223,114	169,666	245,035	$780,378
18	220,902	222,978	81,514	172,131	$697,525
19	119,250	115,223	93,381	160,158	$488,012
20	108,313	169,907	101,350	187,951	$567,521
Total	4,872,450	5,494,985	5,307,029	5,420,152	$21,094,616

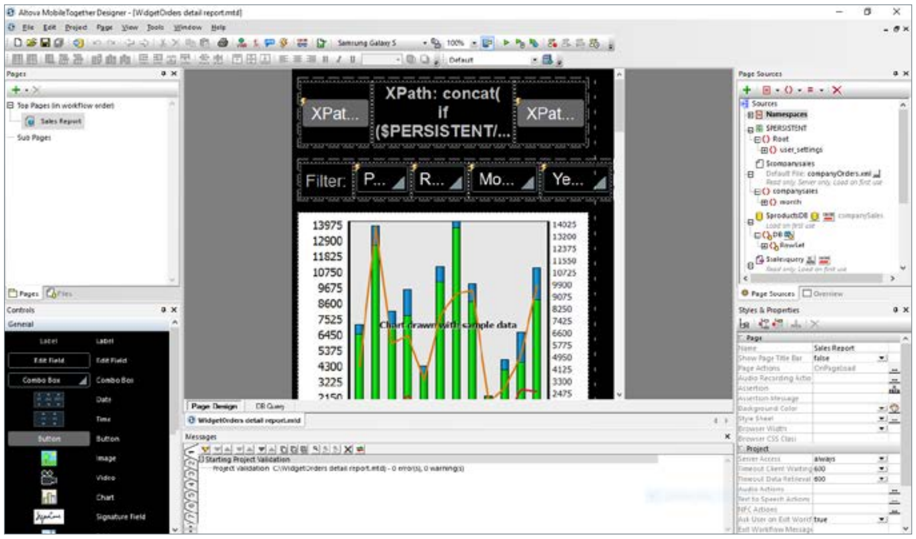
These screenshots show portrait and landscape orientations of the same Company Sales Report app running on an iPhone.

Specific User Interface Features

This section describes in detail the implementation of specific user-interface features, with screenshots from the Company Sales Report and other Altova demo apps.

The MobileTogether Designer

Shown below is a full view of the main Designer window in the center and various helper windows on the left and right. A reference mobile device is selected in the toolbar at the top center of the image.


Altova MobileTogether Designer - [WidgetOrders detail report.mtd]
File Edit Project Page View Tools Window Help
Samsung Galaxy S
100%
Pages
Top Pages (in workflow order)
Sales Report
Sub Pages
XPath: concat(if ($PERSISTENT/...
XPat...
XPat...
Filter: P... R... Mo... Ye...
Chart drawn with sample data
Page Design
DB Query
Messages
Controls
General
Label
Edit Field
Combo Box
Date
Time
Button
Image
Video
Chart
Signature Field
Page Sources
Styles & Properties

Starting Project Validation
Project validation C:\WidgetOrders detail report.mtd - 0 error(s), 0 warning(s)

The same app layout and design applies to all devices, and you can change the reference device or orientation at any time. The examples that follow look in detail at individual features of the app, both in the Designer and on various mobile devices, and how they are configured in the Designer helper windows.

Single Page Design

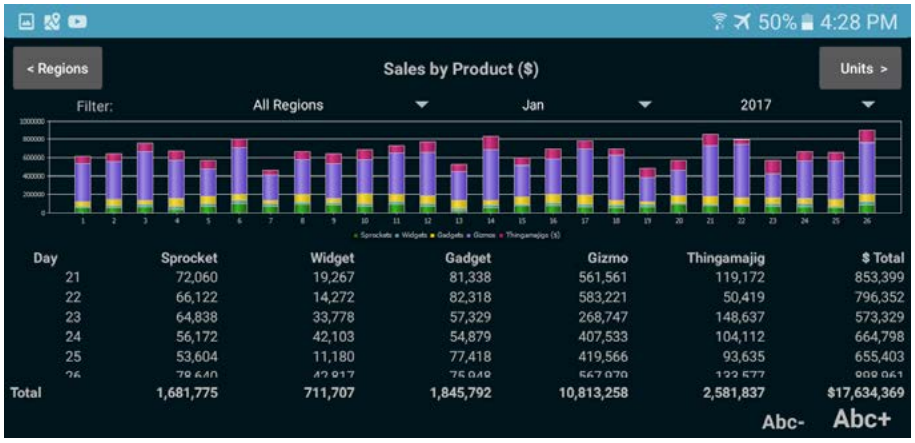
It would be inefficient and cumbersome for any developer to create a separate page for each of 44 different views available in the Company Sales Report. Instead, the entire app is designed in a single layout with various properties assigned to each element that determine what the end-user sees at any moment during execution. As any end-user manipulates controls built into the app, the current view and the contents of charts and data tables are determined by the values of internal variables and the parameters of queries into the underlying sales database.

Navigation buttons within the design simulate the look and experience of multiple pages and allow the user to switch Regions and Products main views and Units and Dollars data views.

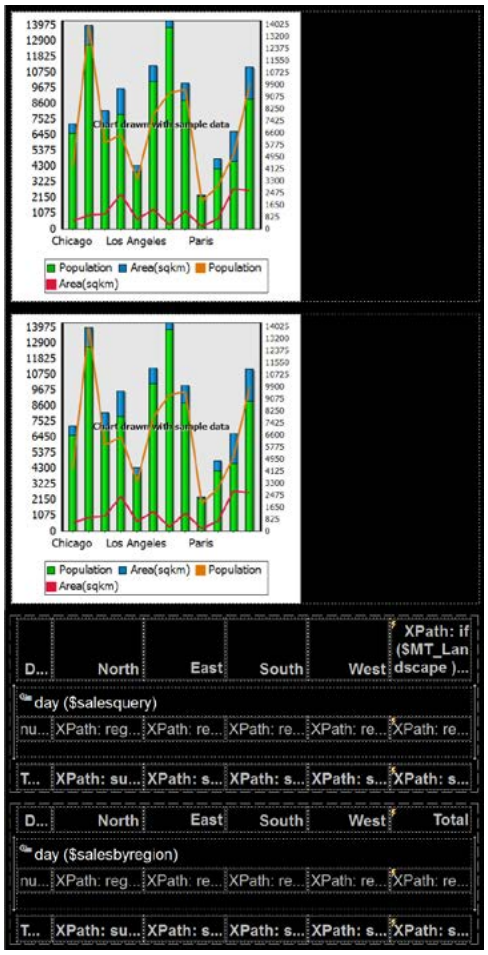
Of course MobileTogether also lets you develop complex apps containing multiple top pages and sub-pages with completely different layouts, with content appropriate for numerous diverse tasks, and many options for user navigation between pages. For example, the Parcel Delivery example app uses separate pages to review delivery assignments, navigate to the next stop, to actually deliver the package and collect the addressee's signature, and to report on the completed run. (The Parcel Delivery app is one example provided with the MobileTogether Designer and can be executed on the MobileTogether Demo server.)

Visibility Property Controls What the End-User Sees

The main body of the Company Sales Report app is a combined view of a chart and data table representing daily sales by product or region for one month:



Day	Sprocket	Widget	Gadget	Gizmo	Thingamajig	$ Total
21	72,060	19,267	81,338	561,561	119,172	853,399
22	66,122	14,272	82,318	583,221	50,419	796,352
23	64,838	33,778	57,329	268,747	148,637	573,329
24	56,172	42,103	54,879	407,533	104,112	664,798
25	53,604	11,180	77,418	419,566	93,635	655,403
26	78,640	42,817	75,948	567,070	133,577	898,061
Total	1,681,775	711,707	1,845,792	10,813,258	2,581,837	$17,634,369

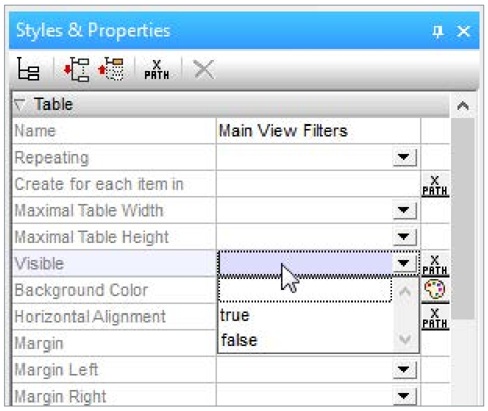


D...	North	East	South	West	XPath: if ($MT_Landscape)...
day ($salesquery)					
nu...	XPath: reg...	XPath: re...	XPath: re...	XPath: re...	XPath: re...
T...	XPath: su...	XPath: s...	XPath: s...	XPath: s...	XPath: s...

D...	North	East	South	West	Total
day ($salesbyregion)					
nu...	XPath: reg...	XPath: re...	XPath: re...	XPath: re...	XPath: re...
T...	XPath: su...	XPath: s...	XPath: s...	XPath: s...	XPath: s...

In the MobileTogether Designer view, the main data section of the app contains a series of chart and table definitions, as seen in this partial view.

Every control, chart, and table in the mobile app design has an individual visibility property that determines whether the element is rendered on the device at any moment during execution of the app. By default, every component added to the app layout will be visible and the property can be left undefined, as seen in the properties helper window illustration here:


Styles & Properties

Table

Name	Main View Filters
Repeating	
Create for each item in	
Maximal Table Width	
Maximal Table Height	
Visible	
Background Color	
Horizontal Alignment	true
Margin	false
Margin Left	
Margin Right	

The end user view of charts and data tables in the Company Sales Report app is determined by the visibility property assigned to each chart and table.

In addition to the default and drop-down menu options shown above, developers can assign visibility via an XPath expression. For instance, here is the XPath expression assigned to the visibility property of first chart shown in the Design View above:

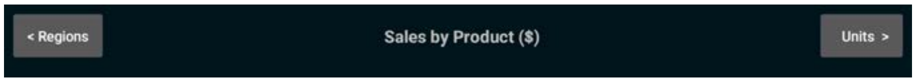
$PERSISTENT/Root/user_settings/mainView = 'Regions' and
$PERSISTENT/Root/user_settings/chartView = 1

The expression tests the values of the same data elements set by the Regions / Products Button and the Dollars / Units Button described above. If the expression evaluates as true, the chart is displayed. If the expression is false, the chart is not displayed.

The combination of two charts and two data tables in the design view corresponds to the chart for sales by region shown in units or in dollars.

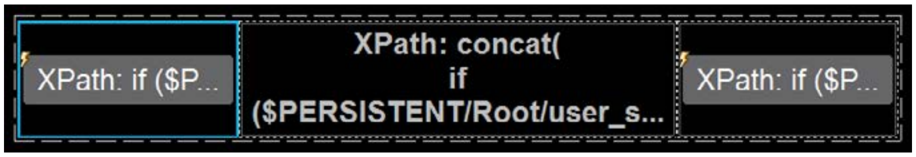
Buttons that Adapt

The top row of the Company Sales Report contains a title in the center and navigation buttons on each side. The button on theleft sets the main view to sales by region or sales by product, and the button on the right sets the values to units or dollars.



These two buttons are only defined once in the app, and their text labels and control actions change based on expressions that set the current view.

Here is the definition of the top row of elements as seen in the MobileTogether Designer with the button on the left selected in the main design window:

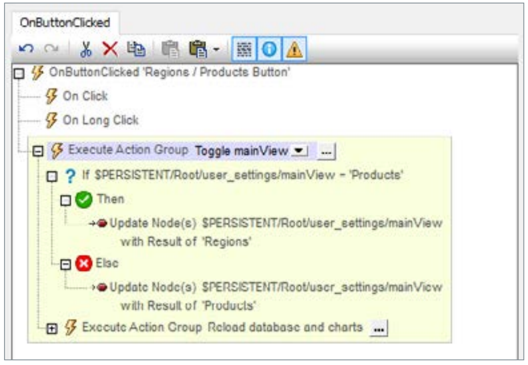


The text in each element is defined by an XPath expression. The button on the left is named the Regions / Products Button in the Properties helper window in the Designer. Here is the expression for the Regions / Products Button:

if ($PERSISTENT/Root/user_settings/mainView ='Products') then
'< Regions' else '< Products'

An element in the persistent data tree holds the value that sets the main view (more about persistent data later). This data element is a string value that is also used to build the main title in the center of the top row.

The button on the right is named Dollars / Units Button and uses a similar data element and XPath expression to set the detail view to dollars or units.



```
OnButtonClicked
OnButtonClicked 'Regions / Products Button'
    On Click
    On Long Click
    Execute Action Group  Toggle mainView
        If $PERSISTENT/Root/user_settings/mainView = 'Products'
            Then
                Update Node(s)  $PERSISTENT/Root/user_settings/mainView
                  with Result of 'Regions'
            Else
                Update Node(s)  $PERSISTENT/Root/user_settings/mainView
                  with Result of 'Products'
        Execute Action Group  Reload database and charts
```

The operation of each button is set in the MobileTogether Actions dialog. Here is a view of the Actions definition for the Regions / Products Button.

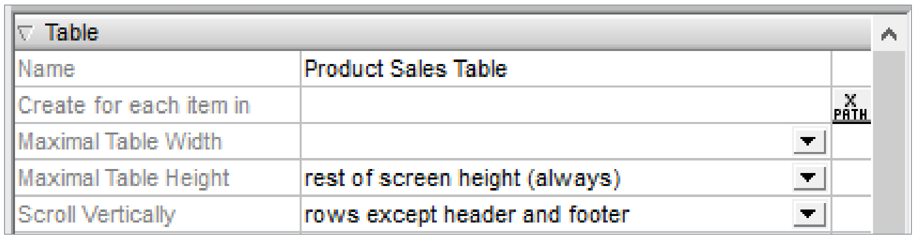
We can see that the same element used to set the button text is manipulated by the button action. When any end user on any mobile device clicks the Regions / Products Button, the value of data element is toggled and views of all charts and data tables are reloaded.

Using XPath to set the button text and action groups to define the button action allows developers to create mobile controls with chameleon-like properties that change during execution of the app. Creating fewer controls with adaptable features accelerates completion of the original app and simplifies revisions and enhancements later on.

Single Layout for Portrait or Landscape Orientation

Several MobileTogether features let us extend the single page design for both portrait and landscape orientations.

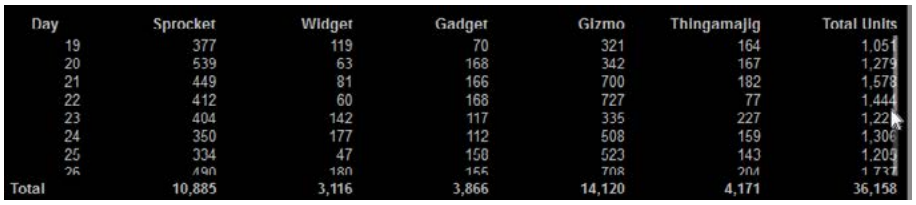
MobileTogether supports scrollable tables that are defined in the Table Properties section. The Company Sales Report main data table has locked headers and footers, and the main body is defined to automatically fill the rest of the screen height with vertical scrolling of the table body.



Table

Name	Product Sales Table
Create for each item in	
Maximal Table Width	
Maximal Table Height	rest of screen height (always)
Scroll Vertically	rows except header and footer

Here is a view of a scrollable table in the Simulator window of the MobileTogether Designer, with the cursor arrow indicating scrolling:



Day	Sprocket	Widget	Gadget	Gizmo	Thingamajig	Total Units
19	377	119	70	321	164	1,051
20	539	63	168	342	167	1,279
21	449	81	166	700	182	1,578
22	412	60	168	727	77	1,444
23	404	142	117	335	227	1,22[illegible]
24	350	177	112	508	159	1,306
25	334	47	158	523	143	1,205
26	490	180	155	708	204	1,737
Total	10,885	3,116	3,866	14,120	4,171	36,158

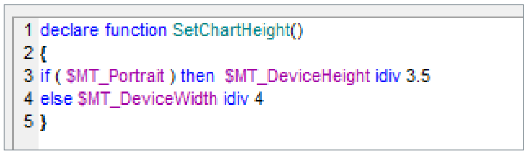
If you want to provide alternate designs or screen layouts for portrait and landscape orientations, the $MT_Portrait and $MT_Landscape system variables let you specify visibility properties based on the current end-user device orientation.

The apps Mortgage Calculator and WidgetsSalesData on the MobileTogether Demo server both illustrate alternative presentations based on orientation.

Size Charts Appropriately for Every Device

The size and width of the main chart are set by a single function using MobileTogether system variables $MT_DeviceHeight and $MT_DeviceWidth that hold the screen dimensions for any specific end user device. Leveraging these variables lets the developer automatically size the chart for each end-user device, even devices released in the future with new dimensions.

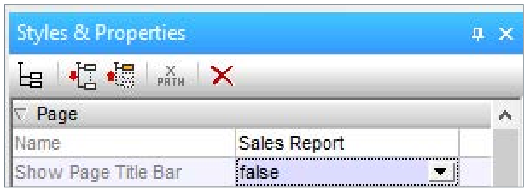
We wanted to adjust the ratio of table and chart heights in landscape mode to allow more room for the main data table. The $MT_Portrait and $MT_Landscape system variables each hold a value of true or false and let the developer test the orientation of the end-user device.
The SetChartHeight() user function defined by the developer takes into account both the dimensions and orientation of the end user device to calculate the final chart size on the fly.



```
declare function SetChartHeight()
{
if ( $MT_Portrait ) then  $MT_DeviceHeight idiv 3.5
else $MT_DeviceWidth idiv 4
}
```

Maximize Data Area

MobileTogether Page Properties allows the developer to hide the page title bar that is specific for each mobile platform.



Name	Sales Report
Show Page Title Bar	false

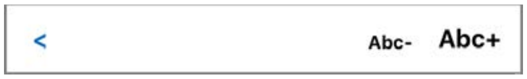
We used this feature in the Company Sales Report app to allow more screen area for data display.

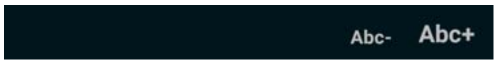
Device-Specific Visibility

Removing the page title bar adds a new complication. The page title bar includes a back button, frequently represented as a left angle bracket character (<) that is not otherwise available on some mobile platforms. Android devices, Windows phones, and Web browsers all have a built-in back button, but iPhones and iPads do not. (Don't worry about remembering these platform-specific intricacies – you will spot any issues when running your app through the MobileTogether simulator for each OS.)

We can fix this by creating a back button with a visibility property for iOS or Windows only, and let other users rely on the built-in version. MobileTogether includes system variables that let the developer identify the end-user platform and specify platform-specific behavior.

If we place this button in a table with columns that size automatically, Android users will not even see a vacant space.





Width of Table Columns

MobileTogether gives the developer multiple options to set the width of each column of a table. Column widths can be:

- a percentage of the device width
- a specific number of pixels
- set to wrap or fill based on text in the column
- allowed to be set automatically based on the number of columns and remaining space available

For instance, the Products data table in the Company Sales Report contains six columns. We could leave the column width properties all blank and the columns would each size automatically to one-sixth of the device width, which is approximately 16.67%.

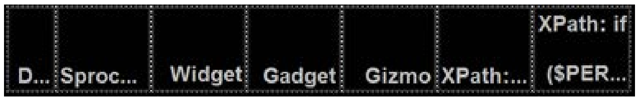
Looking at the column heading for the Products table shows the first column for the date will always contain the shortest text. Therefore, we can define the first column width to be only 8% of the total, allowing the remaining columns to divide the remainder. We only need to set the column width property for the first column and leave the others blank. This increases the width of columns 2-6 to approximately 18.4% each.

That may not seem like a large amount, but more even distribution of blank space between data columns makes the data much more legible.

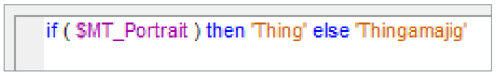
Alternate Column Headings by Orientation

Sometimes the widest element in a column is the header. The products table in the Company Sales Report app uses product names for column headings and Thingamajig is both the longest product name and requires a wider column than the value entries below it.

To better fit small devices in portrait orientation, you can define different headings for portrait and landscape orientation. Here is the table heading as seen in the MobileTogether Designer:



Columns two through five contain product names as plain text. Column six for Thingamajigs is defined as an XPath expression. It an end user device is held in portrait orientation, the shorter heading is displayed, otherwise the full product name will be used.



```
if ( $MT_Portrait ) then 'Thing' else 'Thingamajig'
```

Introduction Page On/Off

Users running a new app for the first time or returning to apps used infrequently may need guidance. Once an app becomes a daily tool, too much help can actually be annoying, wasting time and valuable screen real estate. For instance, a busy executive might run the Company Sales Report app multiple times every day for an update on incoming orders and would quickly tire of an introductory explanation page.

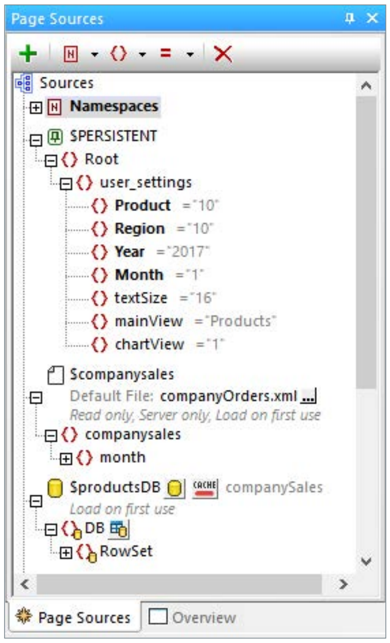
Developers at Altova resolved this conflict by creating introduction pages for each demo app that appear at the top of the page hierarchy and display first by default when the app is run. Each page includes a check box labeled “Skip this intro in the future.” If the end user checks the box, the intro page will not be displayed next time the app is launched on that device. Instead, the second top page in the hierarchy is displayed.

This is explained in the section below.

Persistent Data vs. Workflow Data

You can choose to store data persistently on each client device. This is a good way to preserve settings that will be specific to each user. This data is stored on the client device and reused even for different sessions that are hours or days apart.

Workflow data is not normally preserved on the client device. Instead, workflow data can be pre-defined, calculated on the fly, or read from data sources such as relational databases, Web services, or XML, HTML, or JSON data structures.



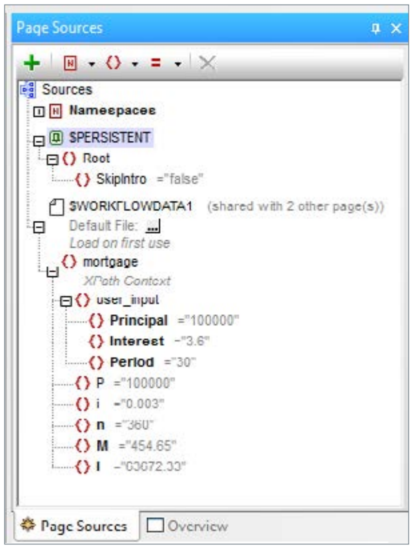
```
Page Sources
Sources
  Namespaces
  $PERSISTENT
    Root
      user_settings
        Product   ="10"
        Region   ="10"
        Year   ="2017"
        Month   ="1"
        textSize   ="16"
        mainView   ="Products"
        chartView   ="1"
  $companysales
  Default File: companyOrders.xml
  Read only, Server only, Load on first use
    companysales
      month
  $productsDB   companySales
  Load on first use
    DB
      RowSet
Page Sources   Overview
```

Shown here is the data source tree for the Company Sales app. The user settings for filter selections, views, and text sizes are stored as persistent data, so each user's individual preferences and last selected view are restored when the app is re-launched.

The default values assigned to persistent elements are applied the very first time the app is run on a new device.

You can examine the Page Sources for MobileTogether sample apps to see other treatments of persistent data.

In the MortgageCalc sample app shown here, and in many others, the SkipIntro element is persistent on the client device and determines whether the introduction page is displayed.



```
Page Sources
Sources
  Namespaces
  $PERSISTENT
    Root
      SkipIntro   ="false"
  $WORKFLOWDATA1   (shared with 2 other page(s))
  Default File:
  Load on first use
    mortgage
    XPath Context
      user_input
        Principal   ="100000"
        Interest   ="3.6"
        Period   ="30"
      P   ="100000"
      i   ="0.003"
      n   ="360"
      M   ="454.65"
      I   ="63672.33"
Page Sources   Overview
```

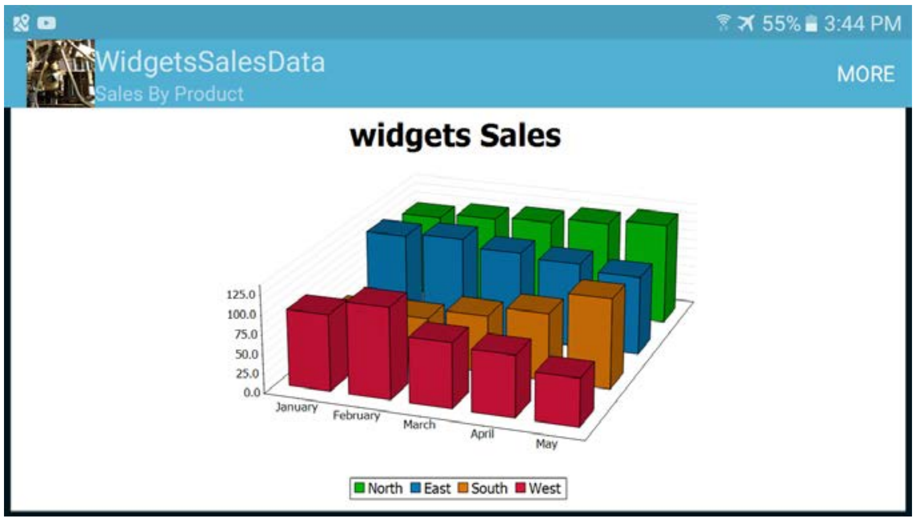
All other elements required to calculate mortgage payments are set to default values initiated each time the app is launched on any device.

Select the Right Chart Style to Represent the Data

The Company Sales Report app uses stacked bar charts to illustrate daily sales by product and region. The stacked bar chart is only one of many chart types available in MobileTogether. MobileTogether apps can also display pie charts, line charts, area charts, candlestick charts and gauge charts, many available in 3-D and tilted variations.

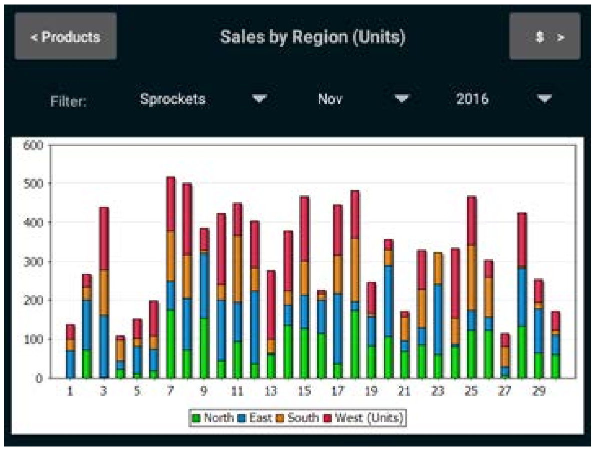
All chart types are fully customizable. You can even use an image to fill a chart background or create overlays of different chart types. For instance, you could overlay a line chart showing a trend line over a bar chart showing daily sales volume.

Every element within each chart is fully also customizable. Various MobileTogether demo apps illustrate chart features, including BizBudget, ChartsDemo, WidgetSalesData, WorldPopulation, and others.



Native Device Colors for Charts

The default chart background for all devices is white with appropriately shaded text and grid line colors. On an Android device, the default colors look like this.



However, any chart property can be set by an XPath expression. You can use MobileTogether global variables that identify specific colors for each device to override chart defaults. The button labelled Dynamic XPath Settings at the top of each chart configuration dialog opens a new dialog that lets you apply XPath expressions to override chart specifications normally defined in the All Settings screens:

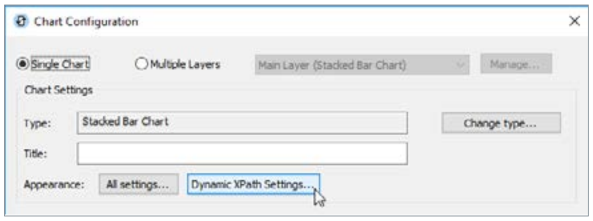


Chart Configuration

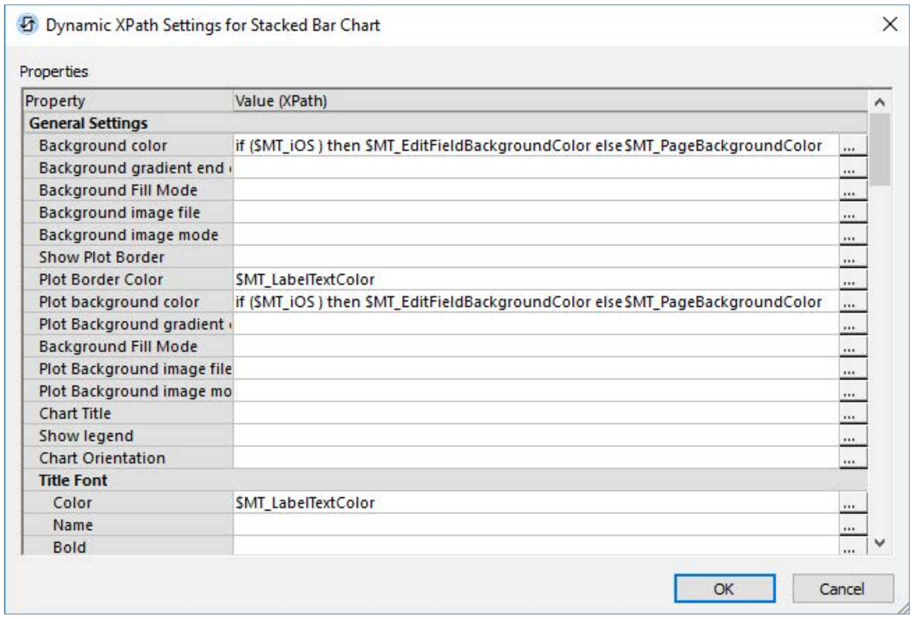
[✓ Single Chart] Multiple Layers Main Layer (Stacked Bar Chart) Manage...

Chart Settings

Type: Stacked Bar Chart

Title:

Here is a partial view of the XPath settings with the values we applied for all the app screenshots except the one directly above that illustrates default chart colors on an Android device:



Dynamic XPath Settings for Stacked Bar Chart

Properties

Property	Value (XPath)
General Settings	
Background color	if ($MT_iOS) then $MT_EditFieldBackgroundColor else $MT_PageBackgroundColor
Background gradient end	
Background Fill Mode	
Background image file	
Background image mode	
Show Plot Border	
Plot Border Color	$MT_LabelTextColor
Plot background color	if ($MT_iOS) then $MT_EditFieldBackgroundColor else $MT_PageBackgroundColor
Plot Background gradient	
Background Fill Mode	
Plot Background image file	
Plot Background image mo	
Chart Title	
Show legend	
Chart Orientation	
Title Font	
Color	$MT_LabelTextColor
Name	
Bold	

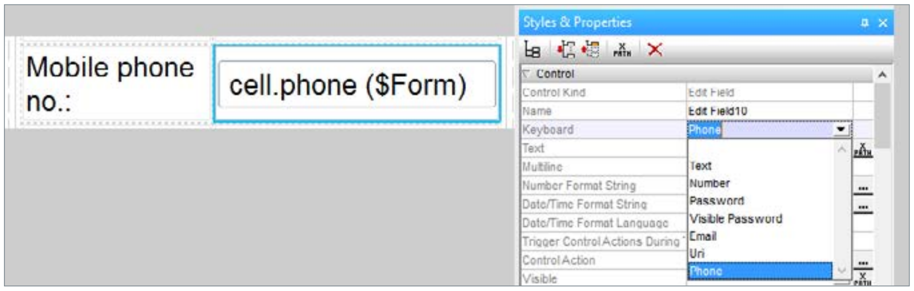
Because of a quirk in device behavior for iOS we had to treat that operating system as a special case. Otherwise we could have simply used $MT_PageBackgroundColor for each color field.

Also, note that this dialog scrolls to access many more fonts and gridline shading options that are all specified as $MT_LabelTextColor for the Company Sales App.

Use the Right Keyboard to Collect User Input

MobileTogether lets developers collect user input via familiar controls like buttons, check boxes, date and time pickers, buttons, and switches. For text input, sometimes the best option is an empty edit field where the user types an entry.

Developers can assign the keyboard for each edit field to prompt the user for the correct data format, as in the mobile phone number edit field shown here in the MobileTogether Designer:

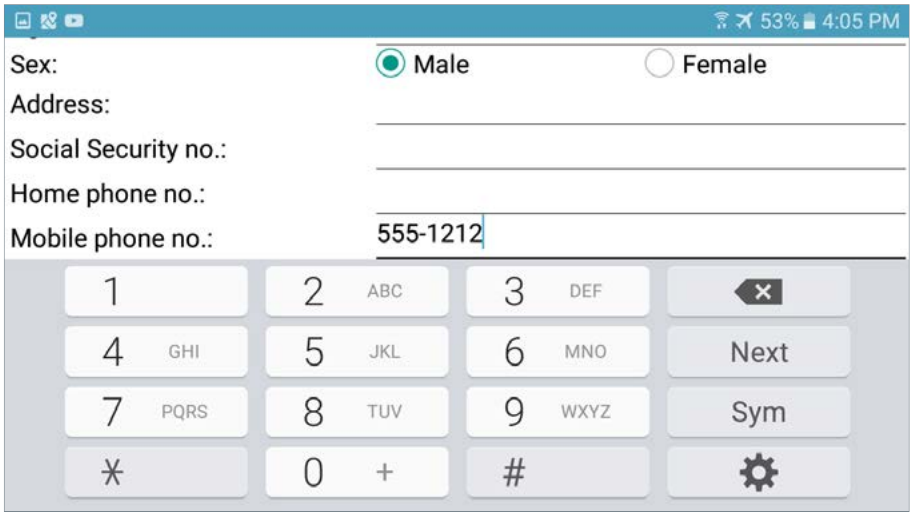


Mobile phone no.: cell.phone ($Form)

Styles & Properties

Control	
Control Kind	Edit Field
Name	Edit Field10
Keyboard	Phone
Text	
Multiline	
Number Format String	
Date/Time Format String	
Date/Time Format Language	
Trigger Control Actions During	
Control Action	
Visible	

Later, when a user running the app selects the Mobile phone entry field, the entry keyboard looks like a telephone keypad:



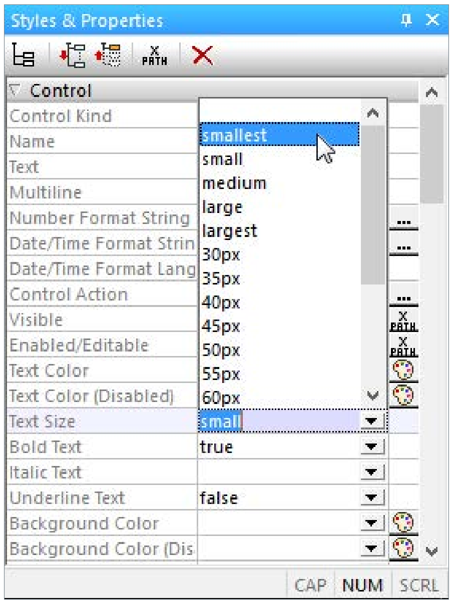
53% 4:05 PM

Sex: [✓ Male] Female
Address:
Social Security no.:
Home phone no.:
Mobile phone no.: 555-1212

The MobileTogether New Patient demo app uses various controls to collect input, including edit fields with several keyboards.
(The New Patient app is one example provided with the MobileTogether Designer and can be executed on the MobileTogether Demo server.)

Let Users Select the Size of Text

MobileTogether offers several strategies for setting text sizes. You can choose a specific pixel height from the menu.



Control

Control Kind	
Name	
Text	
Multiline	
Number Format String	
Date/Time Format Strin	
Date/Time Format Lang	
Control Action	
Visible	
Enabled/Editable	
Text Color	
Text Color (Disabled)	
Text Size	small
Bold Text	true
Italic Text	
Underline Text	false
Background Color	
Background Color (Dis	

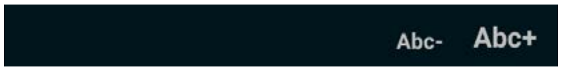
If you choose one of the descriptions from smallest to largest, as shown here, text size is automatically adjusted based on the screen size of the device. Or, you can choose a specific pixel height from the menu.

Even better, you can allow each end user to set the preferred text size for her viewing conditions and device. Doing this is both courteous to users and relieves the developer of the task of determining an appropriate size for each text element to work across all devices.

You can either use two separate buttons or the slider control to allow users to adjust the text size.

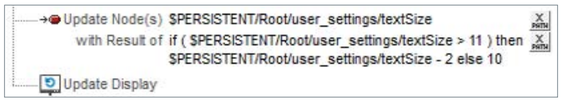
Two Button Text Size

The Company Sales Report uses the two-button technique. The bottom row of controls includes increase and decrease text size buttons with intuitive labels on the right side:



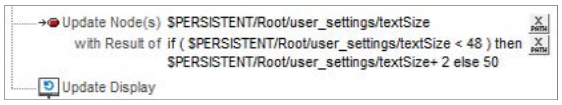
These buttons increase or decrease the value of a persistent data element named textSize from a range of 10 to 50.

Here is the action assigned to the decrease button:



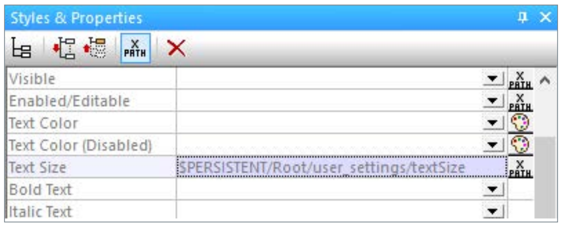
```
Update Node(s)  $PERSISTENT/Root/user_settings/textSize
  with Result of  if ( $PERSISTENT/Root/user_settings/textSize > 11 ) then
                  $PERSISTENT/Root/user_settings/textSize - 2 else 10
Update Display
```

And the action for the increase button:



```
Update Node(s)  $PERSISTENT/Root/user_settings/textSize
  with Result of  if ( $PERSISTENT/Root/user_settings/textSize < 48 ) then
                  $PERSISTENT/Root/user_settings/textSize+ 2 else 50
Update Display
```

Lastly, the text size property for each text element is set by an XPath expression assigning the textSize value, instead of the drop-downmenu selection in the Properties window:

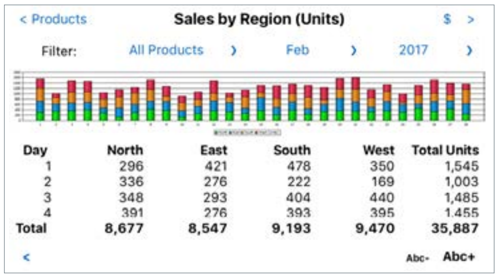


Visible	
Enabled/Editable	
Text Color	
Text Color (Disabled)	
Text Size	$PERSISTENT/Root/user_settings/textSize
Bold Text	
Italic Text	

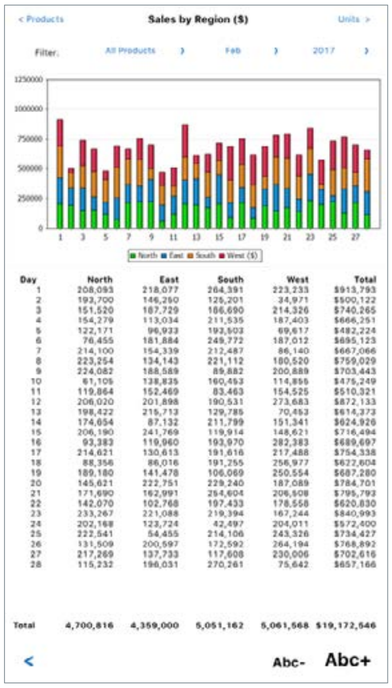
Since the textSize element is defined in the persistent data tree, the user's text size selection is preserved next time the app runs on that device.

This two-button method works well for an app with only a few main pages, like the Company Sales Report, because the user sees the result immediately and can adjust to the preferred size.

Shown here are examples of the Company Sales Report with small text and large text.

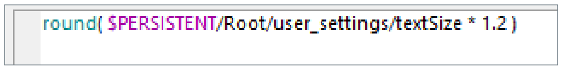


Day	North	East	South	West	Total Units
1	296	421	478	350	1,545
2	336	276	222	169	1,003
3	348	293	404	440	1,485
4	391	276	393	395	1,455
Total	8,677	8,547	9,193	9,470	35,887



Day	North	East	South	West	Total
1	208,093	218,077	264,391	223,233	$913,793
2	193,700	146,250	125,201	34,971	$500,122
3	151,520	187,729	186,690	214,326	$740,265
4	154,279	113,034	211,535	187,403	$666,251
5	122,171	96,933	193,503	69,617	$482,224
6	76,455	181,884	249,772	187,012	$695,123
7	214,100	154,339	212,487	86,140	$667,066
8	223,254	134,143	221,112	180,520	$759,029
9	224,082	188,589	89,882	200,889	$703,443
10	61,105	138,835	160,453	114,856	$475,249
11	119,864	152,469	83,463	154,525	$510,321
12	206,020	201,898	190,531	273,683	$872,133
13	198,422	215,713	129,785	70,453	$614,373
14	174,654	87,132	211,799	151,341	$624,926
15	206,190	241,769	119,914	148,621	$716,494
16	93,383	119,960	193,970	282,383	$689,697
17	214,621	130,613	191,616	217,488	$754,338
18	88,356	86,016	191,255	256,977	$622,604
19	189,180	141,478	106,069	250,554	$687,280
20	145,621	222,751	229,240	187,089	$784,701
21	171,690	162,991	254,604	206,508	$795,793
22	142,070	102,768	197,433	178,558	$620,830
23	233,267	221,088	219,394	167,244	$840,993
24	202,168	123,724	42,497	204,011	$572,400
25	222,541	54,455	214,106	243,326	$734,427
26	131,509	200,597	172,592	264,194	$768,892
27	217,269	137,733	117,608	230,006	$702,616
28	115,232	196,031	270,261	75,642	$657,166
Total	4,700,816	4,359,000	5,051,162	5,061,568	$19,172,546

Note that the main heading is larger than the body text. This is accomplished via an XPath expression for the headline that sets a relationship between the sizes of body and header text:



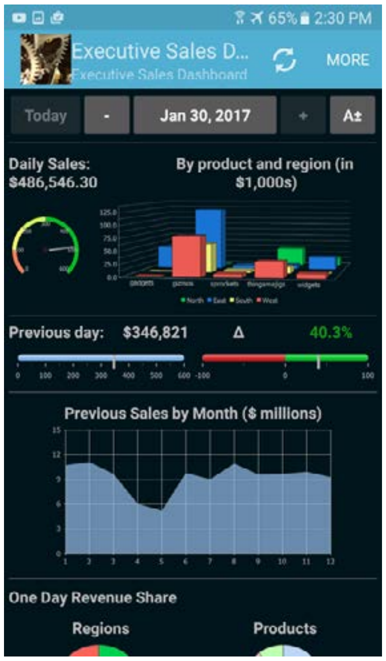
```
round( $PERSISTENT/Root/user_settings/textSize * 1.2 )
```

Changing the multiplier would allow you to set different sizes for various heading and subheading levels in your app.

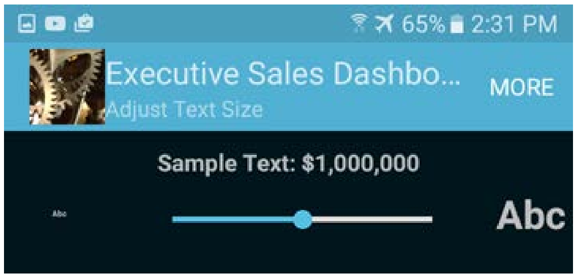
Slider Text Size

Another way to let the user control the text size is by using the slider control. Developers will still want to create a persistent data element for textSize, create user functions to build the size string, and assign the text size property as an XPath expression, as described above in the description of the two-button technique.

But sometimes there is just not enough screen real estate to dedicate two buttons to modify text size. In that case, you can use one button to access a slider control.



The Executive Sales Dashboard app shown left reports sales data for any date in a variety of graphical forms. The row of controls at the top provides a variety of quick date selectors, but only has enough room for one more button to adjust text size.



In this app, the A± button opens a subpage with a slider control to let the user adjust the value of the textSize element.

The sample text on the top line is resized on the fly as the user manipulates the slider control. When the user clicks the back button, the main page is redisplayed with text elements resized. As with the two-button technique, the textSize element is stored as persistent data, and the user's selection will be restored next time the app is launched.

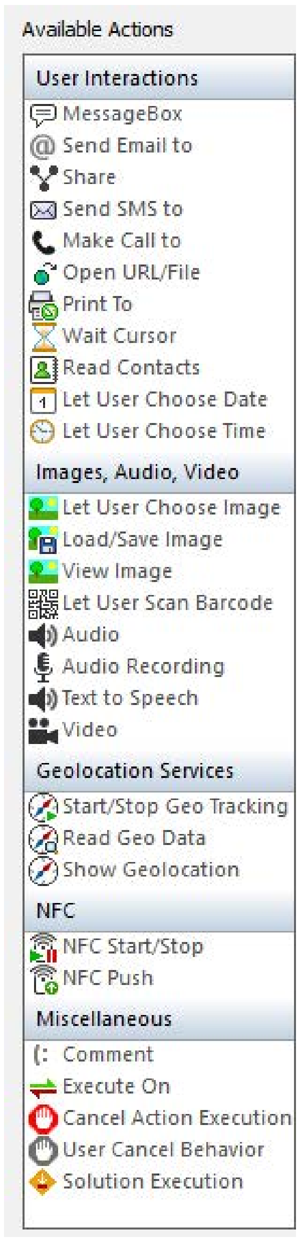
Some apps will have several elements that are appropriate to be saved as user settings, such as inches vs. centimeters, dollars vs. euros, or a default address or image filename. In that case, the developer can build a single preferences page to let the user see and manipulate all the persistent user settings in one place.

Plays Well with Others – Using Helper Apps and Other Mobile Device Features

This guide uses a simple app to illustrate user interface design techniques, but apps built with MobileTogether can do much more than report data stored in backend databases. Developers have access to advanced functionality built into today's mobile phones and tablets. Advanced device features are available in the MobileTogether Actions dialog.

Actions are combined with logic operations into action trees to define complex tasks. For instance, the Expense Reports example app provided with the MobileTogether Designer and available for execution on the MobileTogether demo server implements a full-fledged office workflow application where employees submit expense reports via a mobile solution, with specific roles based on the employee position in the company. For each expense, employees enter expense line-items and attach photos of their receipts. Users can take a new photograph or select an existing one from the device photo library without leaving the Expense report app.

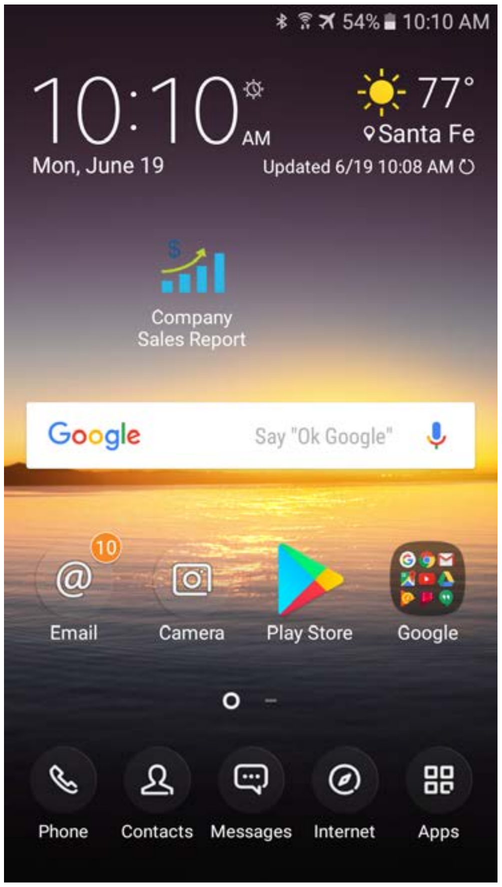
The Share action can be built into a MobileTogether app to let the end user share text and images using the messaging and social-networking apps that are installed on the mobile device. Leveraging common device activities in a familiar way makes mobile apps easier for end users to learn, and helps developers improve favorability ratings among the user community.



Apply a Custom Icon to Your App

The Project Properties helper window includes an entry under Project Properties to assign an image file as an icon to represent the app in the MobileTogether Server apps list or on the mobile device applications screen.

The ideal image format is a 200 x 200-pixel file in .png format. By default, the MobileTogether icon will be used if no image is assigned, but it is much better to choose an image that is easy to recognize and accurately represents app functionality. The Company Sales Report app uses a stylized dollar sign on a contrasting background.



Here is a view of the icon on the home screen of an Android phone.

Next Steps

Software developers or other technical users can start developing elegant cross-platform mobile apps with MobileTogether by following these steps:

- Install the MobileTogether Client app from the app store for your mobile device and run some of the demo apps
- Try out some real-world apps built with MobileTogether
- Download the MobileTogether Designer, run the Quickstart Tutorials in the MobileTogether Designer Help, and examine how the demo apps were built
- Watch MobileTogether YouTube videos and read the Altova Blog for more information
- Build your own app and run it in the MobileTogether Simulator
- Connect your mobile device to the MobileTogether Designer workstation and see your app run live

Install the MobileTogether Server and choose either Instant Deployment or App Store Deployment to distribute your app.

We hope all the MobileTogether functionality described here inspires you to build your own powerful, cross-platform app in record time, with an elegant, intuitive interface that delights users on all mobile devices!

About Altova

Altova® is a software company specializing in tools that assist developers with data management, software and application development, mobile development, and data integration. The creator of XMLSpy® and other award-winning products, Altova is a key player in the software tools industry and the leader in XML solution development tools. The company offers a complete line of desktop developer software for XML, SQL, and UML; high-performance workflow automation server products; and a cross-platform mobile development platform. Altova focuses on its customers' needs by offering a product line that fulfills a broad spectrum of require-ments for software development teams. With over 5 million users worldwide, including 91% of Fortune 500 organizations, Altova is honored to serve clients from one-person shops to the world's largest organizations. Altova is committed to delivering standards-based, platform-independent solutions that are powerful, affordable and easy-to-use. Founded in 1992, Altova is headquartered in Beverly, Massachusetts and Vienna, Austria. Visit Altova on the Web at: https://www.altova.com.