AppBuilder

AppBuilder 1.0

DOCUMENT ID: DC02001-01-0100-01
LAST REVISED: November 2013

Contents

Installing AppBuilder

You can install AppBuilder on Microsoft Windows or Mac OS X systems.

Prerequisites

- You must have the latest version of the Google Chrome browser, or Apple Safari installed.
- AppBuilder is packaged using the NodeJS package module. Therefore, you must install NodeJS onto your workstation to run AppBuilder:
  1. Download the NodeJS installer for your system from *http://nodejs.org/download/*.
  2. Run the NodeJS installer, and follow the prompts in the Install Wizard.
  3. Verify that the node.js folder is in your System Environment PATH.
  4. Open a command prompt, and enter **npm** to verify the NodeJS package manager is installed properly.
- If you are using a proxy server you must configure npm. At the command prompt or terminal window (on Mac), enter:

```
npm config set proxy  <proxy server:port>
npm config set https-proxy <proxy server:port>
```

- If you use a web proxy to connect to the Internet or consume OData services from the Internet, you must set the proxy host and port in the `ide.json` file as discussed in *Setting Up the Development Environment*.

Installing AppBuilder on Windows

1. Extract the appbuilder-1.0.zip file to the directory of your choice.
2. From the directory where you extracted the contents of the zip file, double-click the `run.bat` file to launch AppBuilder.
3. AppBuilder opens in a browser with the URL: `http://127.0.0.1:9009/ide/ares/index.html`.

Installing AppBuilder on Mac

1. Extract the appbuilder-1.0.zip file to the directory of your choice.
2. Open **Terminal**.
3. Change to the directory where you extracted the zip file.
4. Enter `chmod +x *.sh` at the prompt to grant execute permissions to the AppBuilder start shell scripts.
5. Enter `./run.sh` at the prompt to launch AppBuilder.
6. AppBuilder opens in a browser with the URL: `http://127.0.0.1:9009/ide/ares/index.html`.

Setting Up the Development Environment

Follow the additional steps to set up your AppBuilder environment.

Setup Proxy

If you install AppBuilder on a machine that uses web proxy to connect to the Internet and consume OData services, you need to modify the `ide.json` file.

1. Open the `ide.json` file in `<AppBuilder_Home>\ares-project\`
2. Set the proxy host. For example:

```
"httpProxy": {
    "host":"proxy.sin.sap.corp",
    "port":8080,
    "upBase64":"",
    "user":"",
    "password":",
}
```

3. Save `ide.json` and exit.

Developing with Cordova

AppBuilder enables you to create an Apache Cordova project from your AppBuilder project. You can also add Kapsel plugins to your Cordova project. Kapsel, a component of SAP® Mobile Platform SDK, leverages the Cordova application container and provides SAP plugins to make the Cordova container enterprise-grade, allowing it to more seamlessly integrate with SAP Mobile Platform Server. The Kapsel plugins provide capabilities like application life cycle management, implementation of a common logon manager and single sign-on (SSO), integration with SAP Mobile Platform Server-based push notifications, and so on.

Prior to deploying your AppBuilder applications to Kapsel on SAP Mobile Platform, the following prerequisites must be met:

- Install SAP Mobile Platform Server 3.0 and have it running in your local environment
- Install SAP Mobile Platform SDK 3.0 with the Kapsel SDK selected
- Install Apache Cordova Command Line Interface (CLI)
- Install Kapsel Command Line Interface (CLI)

For information about installing and setting up SAP Mobile Platform Server and SAP Mobile Platform SDK, see specific documentation for *SAP Mobile Platform Server* and *SAP Mobile Platform SDK*.

To install the Apache Cordova Command Line Interface:

1. Install Git according to the instructions found here: *http://git-scm.com/book/en/Getting-Started-Installing-Git*.

Note: If you are using a proxy server, you must configure git At the command prompt enter:

For Windows:

```
git config --global http.proxy <proxy server:port>
git config --global https.proxy <proxy server:port>
```

For Mac:

```
sudo git config --global http.proxy <proxy server:port>
sudo git config --global https.proxy <proxy server:port>
```

2. Install the Cordova CLI:
   a. Open a command prompt and enter.
      On Windows: `npm install -g cordova@<latest_supported_version>`
      On Mac:`sudo npm install -g cordova@<latest_supported_version>`
      `-g` indicates that Apache Cordova should be installed globally.

      Note: If you are installing on Mac and you see a warning message that you are installing globally into a root-only directory, run this command to change the owner of the command line interface installation foler:

      ```
      sudo chown -R user_name /usr/local/lib/node_modules/cordova
      ```

      You can copy the command text from the error message and paste it at the command prompt at the bottom of the terminal window.
   b. On Mac, when prompted, enter your root user password.
   c. Verify the Cordova installation. At the command prompt enter:

      ```
      cordova -v
      ```

To install the Kapsel CLI:

You must have SAP Mobile Platform SDK with the Kapsel SDK installed locally to install the Kapsel CLI. Follow these steps for installation:

1. Open a command prompt or Terminal and enter.
   On Windows: `npm install -g C:\<SAPMobileSDK_Home>\MobileSDK3\Kapsel\CLI`
   On Mac:`sudo npm install -g C:\<SAPMobileSDK_Home>\MobileSDK3\Kapsel\CLI`

   Note: If you are installing on Mac and you see a warning message that you are installing globally into a root-only directory, run this command to change the owner of the command line interface installation foler:

   ```
   sudo chown -R user_name /usr/local/lib/node_modules/cordova
   ```

You can copy the command text from the error message and paste it at the command prompt at the bottom of the terminal window.

2. Verify the Kapsel CLI installation. At the command prompt enter:

```
kapsel package
```

Setting Up the Mac Environment

If you are developing on Mac, you must install the following software (if not already installed).

- Xcode and Xcode Command Line Tools installed (to deploy to iOS platform)
- ios-sim (to allow the Cordova command line to start the iOS simulator on Mac)

Note: On Mac, the latest OS X is supported.

Install Xcode Command Line Tools

To install the Xcode Command Line Tools on Mac:

1. Start Xcode.
2. Navigate to **Xcode** > **Preferences**.
3. Select **Downloads**.
4. Click **Install** next to Command Line Tools if it is not already installed.

Install ios-sim

To install ios-sim on Mac:

1. Download the ios-sim tool files from *https://github.com/phonegap/ios-sim*.
2. Open **Terminal** and enter: `sudo npm install -g ios-sim`
3. When prompted, enter your root user password.
4. Verify the ios-sim installation by entering this command in the terminal window: `ios-sim --version`. The output displays the ios-sim version installed.

Developing for the Android Platform

If you would like to the Android Platform, there is additional software that must be included in your development environment:

- Download and extract Apache ANT (current version), and add it to the system variable path: `PATH=%PATH%; C:\apache-ant-<version>\bin`. See *http://ant.apache.org* for more information.
- Android SDK 4.1.2 or later
- Google USB Driver - download from *http://developer.android.com/sdk/win-usb.html*. Google USB Driver is an optional Android SDK component that you need only if you are developing on Windows and want to connect a Google Android-powered device to your development environment over USB.

Install Android SDK

To install the Android SDK:

1. Confirm that your system meets the requirements at *http://developer.android.com/sdk/requirements.html*.
2. Download and install the supported version of the Android SDK starter package.
3. Add the Android SDK to your PATH environment variable:

   On Windows, add `<Android SDK Location>\tools` to the PATH environment variable.

   On Mac, open a terminal window and enter `export PATH=$PATH:<path to Android SDK>/tools`.
4. Launch the Android SDK Manager and install the Android tools (SDK Tools and SDK Platform-tools) and the Android API.
5. Launch the Android Virtual Device Manager, and create an Android virtual device to use as your emulator.

Launching AppBuilder

Launch AppBuilder in your browser.

Prerequisites

- You must have the latest version of the Google Chrome browser, or Apple Safari installed.
- On Mac, ensure that you have granted execute permissions to the AppBuilder start shell scripts as described in *Installing AppBuilder*.
- Complete the steps listed in *Setting up the Development Environment*.

Launching AppBuilder on Windows

To launch AppBuilder from the command line:

1. Open a command prompt
2. Change to the directory where you extracted the contents of the AppBuilder `.zip` file.
3. Execute `run.bat`.

Note: If you prefer not to use the command line, you can launch AppBuilder by navigating to the file where you extracted the contents of the AppBuilder `.zip` file and double-clicking on `run.bat`

The AppBuilder Start page is displayed in a browser window with the following URL: `http://127.0.0.1:9009/ide/ares/index.html`.

Installing AppBuilder on Mac

Launch AppBuilder using **Terminal**.

1. Open **Terminal**.
2. Change to the directory where you extracted the zip file.
3. Enter `./run.sh` at the prompt to launch AppBuilder.

The AppBuilder Start page is displayed in a browser window with the following URL: `http://127.0.0.1:9009/ide/ares/index.html`.

AppBuilder Overview

AppBuilder is an integrated development environment with an open architecture built with standard technology, namely HTML5, JavaScript, and OData services.

AppBuilder is browser-based, and includes WYSIWYG features that you can use to create HTML5/JS-based mobile applications. The simple drag-and-drop interface leverages SAP UI5 controls for rapid development, and enables you to use data services from SAP Mobile Platform, OData, REST, or local filesystems. AppBuilder facilitates quick and simple deployment of your mobile application by providing predefined application templates that you can use to create:

- generic applications (simple, chart, or superlist applications)
- tab-based applications

AppBuilder also includes three sample applications that can be used for reference purposes:

- AnalyticSample – demonstrates the charting capabilities and page transitions
- SuperListSample – this sample demonstrates the use of a SuperList
- TabAppSample – shows an application that uses multiple tabs

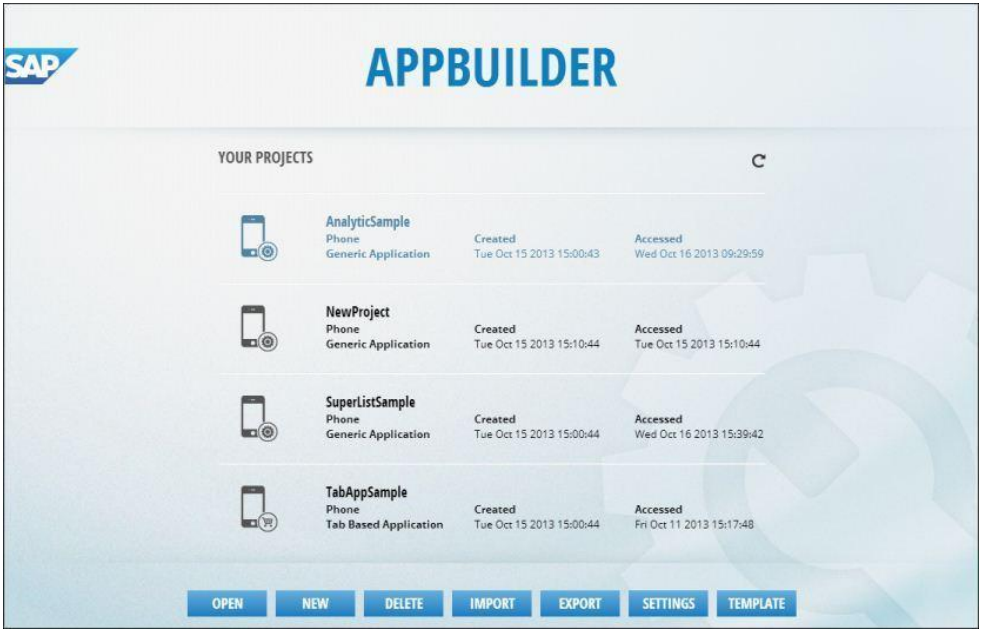
Additional features include:

- A form designer that enables you to design the application form interactively. You can drag and drop controls from the toolbox palette to the form designer canvas and position the controls any way you want. You can also customize your application with themes and styles.
- A code generator that generates essential and skeleton code which are synchronized with your application form so you can switch between source code and design views while designing your application.
- A data source wizard that allows you to connect to the application's data source in design-time, and then bind it to the controls on the form.
- Multiple deployment options that allow you to create a Cordova project, launch to a simulator, publish to a project directory, or deploy to the Kapsel Cordova container on SAP Mobile Platform Server.

Project Management

AppBuilder's interface is designed to provide you with simple and easy project management.

When AppBuilder launches in the browser, the Start Page is displayed. The Start Page shows a list of current projects (including packaged sample projects), and options for managing those projects.



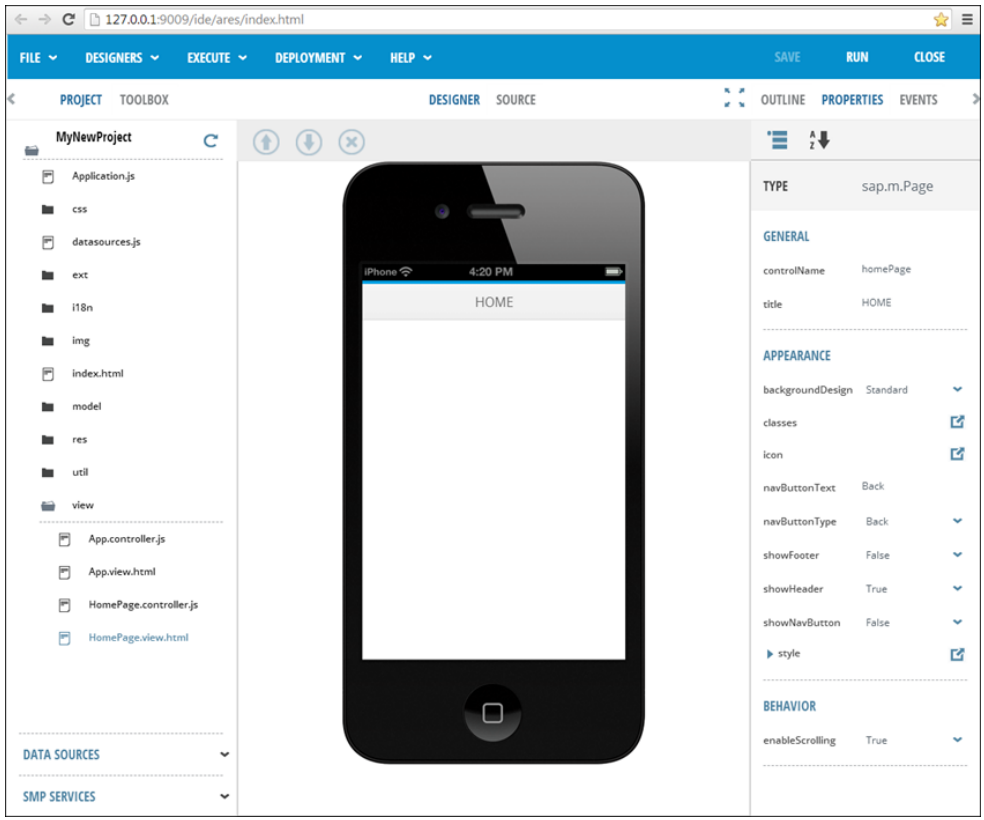
From the Start Page, you can perform simple project management functions:

- **Open** – select the project and click **Open**. You can also open a project by double-clicking the project name.
- **New** – create a New project, specifying the project name, device type for the application, and template you would like to use the application design.
- **Delete** – select the project and click **Delete**. This permanently deletes the project and associated files from your system.
- **Import** – import an existing AppBuilder project into the AppBuilder workspace. This adds the project to your projects list on the start page.
- **Export** – select the project and click **Export** to compress the project into a `.zip` file and saved into your `Downloads` folder.
- **Settings** – edit project settings (name and device type) and configure version and author information for your project. You can also change the assigned theme for your project using this function.
- **Template** – select the project and click **Template** to save the project as a custom template that you can choose when you create new projects.

Navigating the AppBuilder Workspace

AppBuilder's project workspace is designed to provide you with a simple, easy, and highly productive development environment.

When you open a project in AppBuilder, the project page opens in the workspace.



The workspace includes the AppBuilder Menu, Project Explorer, Toolbox and the default Form Designer.

AppBuilder Menu

The AppBuilder menu at the top of the provides you with additional project management functions.

File Menu

The File menu appears provides you with functions for managing the current project you are working with.

- **New File** – allows you to create a new file within the project.
- **New Folder** – allows you to create a new folder within the project for file organization.
- **New Page** – allows you to create additional pages within your application. When you create a new page, can create a general page, a tab page, or a dialogue page. Each page type provides you with different control functions that you can use to configure the layout and behavior for your application.
- **Upload** – allows you to upload additional files to your project.

- **Delete** – allows you to delete a project file or folder.
- **Save** – saves the file or page in its current state.
- **Save As** – saves the file as a new page. This option allows you to create a new page from the current content without having to start from scratch.
- **Close** – exits the project page and returns you to the AppBuilder Start Page.

Designers

The Designers menu allows you to add and configure new chart and superlist controls in your project, and opens the associated designer.

- Chart Designer – a UI within AppBuilder which allows you to add and configure charts within your project. Using the Chart Designer, you can bind two- or three-dimensional data to the chart control and then configure the appearance and behavior of the chart within the application.
- SuperList Designer – a UI within AppBuilder which allows you to create large lists of data within our project. Using the SuperList Designer, you can configure the data to be displayed in list view, form view, or group (grid) view.

Execute

The Execute menu allows you to run your application on a device simulator. When you click **Run**, the application launches in the simulator. From here, you can configure the device (phone or tablet), and orientation (portrait or landscape) to see how your application will be rendered on a device.

Deployment

AppBuilder provides several different deployment options for your projects.

- **Create Cordova Project** – generates a .zip file containing the web content, scripts, and selected plugins for an Apache Cordova project.
- **Launch to Simulator** – builds and launches the application in the simulator for the specified platform (iOS or Android).
- **Publish to Project Directory** – compiles the project and zips it to a project file in your `<AppBuilder_Home>\ares-project\hermes\filesystem\root\guest` directory within AppBuilder.
- **Kapsel on SMP Server** – allows you to specify server settings to deploy the project as a Kapsel application to SAP Mobile Platform Server.
- **Workspace Setting** – allows you to set the directory to which your project will be saved when you select the **Create Cordova Project** deployment option.

Help

The Help menu provides access to AppBuilder documentation, as well as options for configuring AppBuilder and information about your AppBuilder installation:

- **Help Contents** – opens the AppBuilder documentation in another browser tab.

- **Preferences** – opens a dialogue box that allows you to configure the appearance of the code in the Source Designer.
- **SMP Settings** – opens the dialogue box that allows you to create an application definition and security profile so your application can access back-end SAP services. This feature is necessary for applications that are deployed to SAP Mobile Platform Server.
- **About AppBuilder** – provides version information about the installation of AppBuilder that you are working in.

Project Explorer

The project explorer appears in the left pane of the AppBuilder workspace and lists all the directories and files for a project in a tree view.

When you create a new project, many files and folders are automatically generated and placed in `<AppBuilder_Home>\ares-project\hermes\filesystem\root\guest\<Project_Name>`. Within the AppBuilder workspace, the files are displayed in the Project Explorer in the left pane.

Click on the folder names to view the contents of the directories. To view the content of the files, double-click, and the contents of the file will be displayed via the Code Editor in the center pane of the workspace. Once you have opened the code editor, you can return to the Form Builder by opening one of the page view html files (for example, `HomePage.view.html`) in the `view` directory.

Toolbox

AppBuilder's Toolbox contains all of the controls that you can use in your AppBuilder projects.

The controls that are available to you in the AppBuilder Toolbox palette are dependent upon which Designer you are using. The Form Designer and the SuperList Designer allow you to add additional controls to your form. The Form Designer features the full library of controls in the Toolbox palette, and the SuperList Designer includes a subset of those controls. You can configure each of the AppBuilder controls in the Properties panel as described in *Configuring Controls in the Project*.

Form Designer Only Controls

These controls are available in the Form Designer palette:

- Bar – a simple control for aesthetic use.
- Button – a simple push button used for initiating actions or events.
- BusyIndicator – a display control that can be used to disable single controls. It prevents the user from interacting with the control
- Chart – a graphic control associated with data from a data source. See *Creating a New Chart* for more information about working with Charts using the Chart Designer.
- CheckBox – a decision control. When checked, the associated value is set. If unchecked, the associated value is not set.
- DateTimeInput – an input control that allows the user to select a specific date and time.
- FlexBox – a container control used to provide a flexible layout that optimizes use of available space.
- HBox – a fixed container control used to provide a simple way to place several controls next to each other horizontally, without wrapping.
- HTML – a library control that can be used in another control such as the tab or panel control. This control displays the HTML that you define in the `content` property.
- Image – a wrapper around the HTML tag for graphics. When using this control, you must set the `src` path to where the image file is stored.
- Input – a control that can be used to prompt the user for information. You can configure the type property to indicate what information (for example, password, date and time, text, etc.) to associate the input with.
- Label – a simple control that provides some short explanative text, usually next to a value holder such as a Text control.
- RadioButton – a decision control used to provide the user with a single choice.
- ScrollContainer – a container control with a horizontal only, vertical only, both, or no scrollbars. When you use this control, you must set the `horizontal` and `vertical` properties to indicate which scrollbars to display.

- SegmentedButton – a container control used to display a group of buttons. The buttons behave like toggle buttons, in that only one button can be selected at once. When you use this control, you must configure the properties for each button it includes.
- Select – a decision control that presents a menu of options. Configure the `items` property to set what those options are.
- Slider – a simple interactive control used to check various size settings. This control can be used in conjunction with the Chart control.
- SuperList – a graphical control that allows you to display data from a datasource in list form. See *Creating a New SuperList* for information about working with SuperLists using the SuperList Designer.
- Switch – a customizable toggle button control. When you use this control, you can set the `type` property, and customize the default On/Off text that is displayed on the button.
- Text – a simple control for single-row text entry. You can use this control with other container controls, and when used, you can configure the `text` property to set the text to display.
- TextArea – a simple control for multiple-row text entry. Like the text control, you can use this control with other container controls, and when used, you can configure the `value` property to set the text to display. This control uses scroll bars to accommodate text wrapping behavior.
- VBox – a fixed container control used to provide a simple way to place several controls next to each other vertically, without wrapping.

SuperList Designer Controls

These controls are available in the SuperList Designer palette:

- Button – a simple push button used for initiating actions or events.
- CheckBox – a decision control. When checked, the associated value is set. If unchecked, the associated value is not set.
- GroupBox – a container control used to group other controls.
- Image – a wrapper around the HTML tag for graphics. When using this control, you must set the `src` path to where the image file is stored.
- Input – a control that can be used to prompt the user for information. You can configure the type property to indicate what information (for example, password, date and time, text, etc.) to associate the input with.
- Label – a simple control that provides some short explanative text, usually next to a value holder such as a Text control.
- Panel – a container control which consists of the main content area and a header.
- Select – a decision control that presents a menu of options. Configure the `items` property to set what those options are.
- TextArea – a simple control for multiple-row text entry. Like the text control, you can use this control with other container controls, and when used, you can configure the `value`

property to set the text to display. This control uses scroll bars to accommodate text wrapping behavior.

See also

- *Configuring Controls in the Project* on page 25
- *Adding Controls to the Project* on page 24
- *Creating a New SuperList* on page 37
- *Creating a New Chart* on page 47

Designers

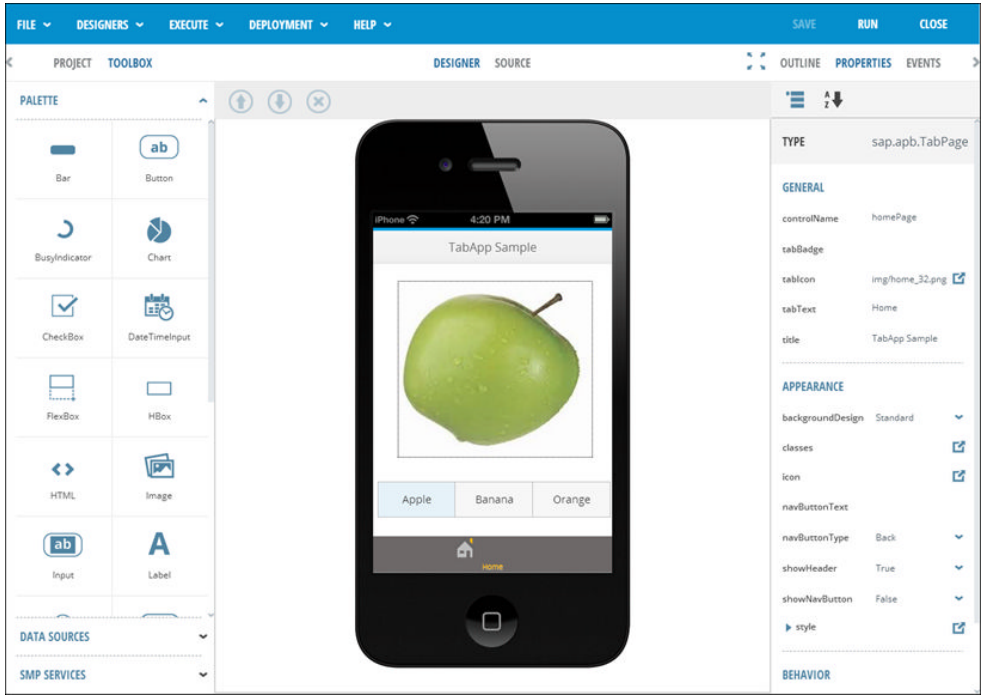
The AppBuilder workspace includes several designers to enable you to develop applications quickly using drag and drop functionality.

The main designer in the AppBuilder workspace is the Form Designer. This is the default designer that displays when you open or create a new project. In addition, AppBuilder includes a Chart Designer and a SuperList Designer that you can use to edit or add chart and superlist controls to a project. The Chart Designer and SuperList Designer both feature interfaces and properties that are specific to their corresponding controls.

Form Designer

The Form Designer is the main workspace in which you build your application, and is the default graphical interface for application development in AppBuilder.

The Form Designer provides all of the tools you need to easily create generic and tab-based applications.

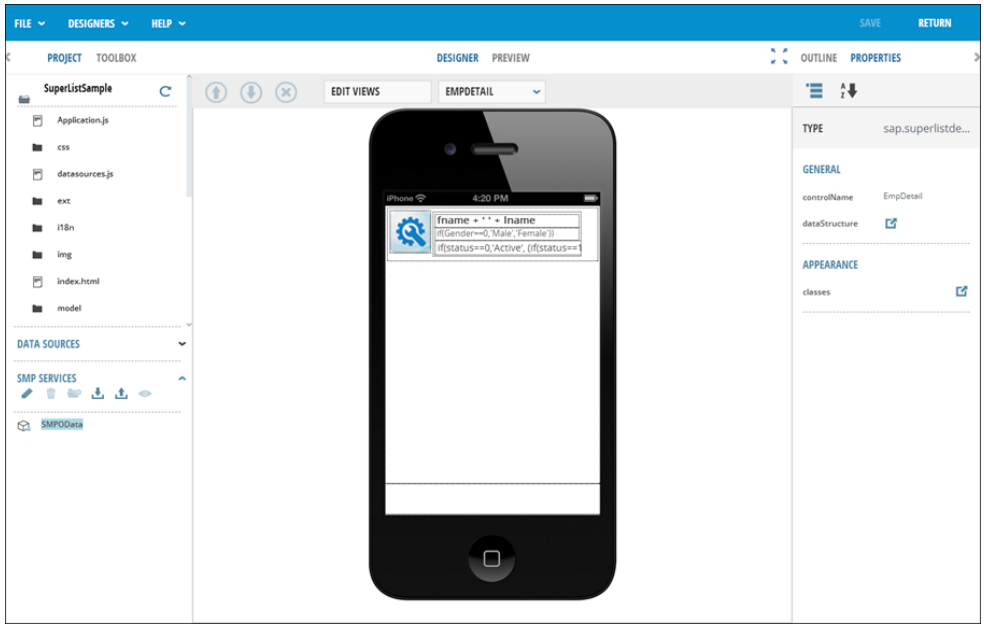


- Toolbox – provides a palette containing all of the controls that available to you based upon the type of application you are designing. Simply drag the desired control from the palette in the left pane and drop it in the desired location on the form in the center pane.
- Data Sources – provides the interface to manage data sources for use with your project. From the Data Sources panel, you can Add, Edit, Delete, Import, Export, and Preview four types of data sources: ODataService, ODataQuery, RESTful, and File.
- SMP Services – provides the interface to manage SAP Mobile Platform services. From the SMP Services panel, you can configure your SMP Server Profile, and then retrieve services from SAP Mobile Platform. You can also manually add services.
- Designer – displays the form in which you can graphically organize your controls.
- Source – displays the source code editor. When you click the **Source** button, the source code for the page you are working in is displayed.
- Outline – displays the relationships between controls for the page you are currently working on.
- Properties – provides the interface to manage and configure properties for your application pages and the controls that you work with. When you select a control that you have placed in the form, the control properties are displayed. You can then modify them as appropriate.
- Events – provides the interface in which you can manage event handling for your application pages as well as the controls in those pages.

SuperList Designer

The SuperList Designer looks similar to the Form Designer, but the Source editor and Events panel are removed, and the Menu bar only displays actions related to superlist functions.

The SuperList Designer allows you take advantage of rich functionality such as create, retrieve, update, and delete (CRUD) operations and navigating master detail views in multiple formats without writing any code.



The SuperList Designer supports three different views:

- ListView – displays data from from your specified data source in list format on your device screen.
- FormView – displays a row of data from your specified data source on your device.
- GroupView – displays data from your data source grouped by the column that you specify.

When you click the **Edit Views** button, the View Manager appears offering you options to add, update, and delete views, as well as change the order of the views. The SuperList control is metadata-based and data source-independent. It provides easy-to-use APIs that allow you to retrieve data from different data sources.

Chart Designer

The Chart Designer looks similar to the Form Designer, but the Toolbox, Source editor, Events panel, and Outline panel are removed, and the Menu bar only displays actions related to chart control functions.

You can access the Chart Designer in one of these two ways:

- From the AppBuilder Form Designer menu, choose **Designers** > **New Chart** .
- To open the Designer for an existing chart, in Project Explorer double-click the chart's `.mameta` file.

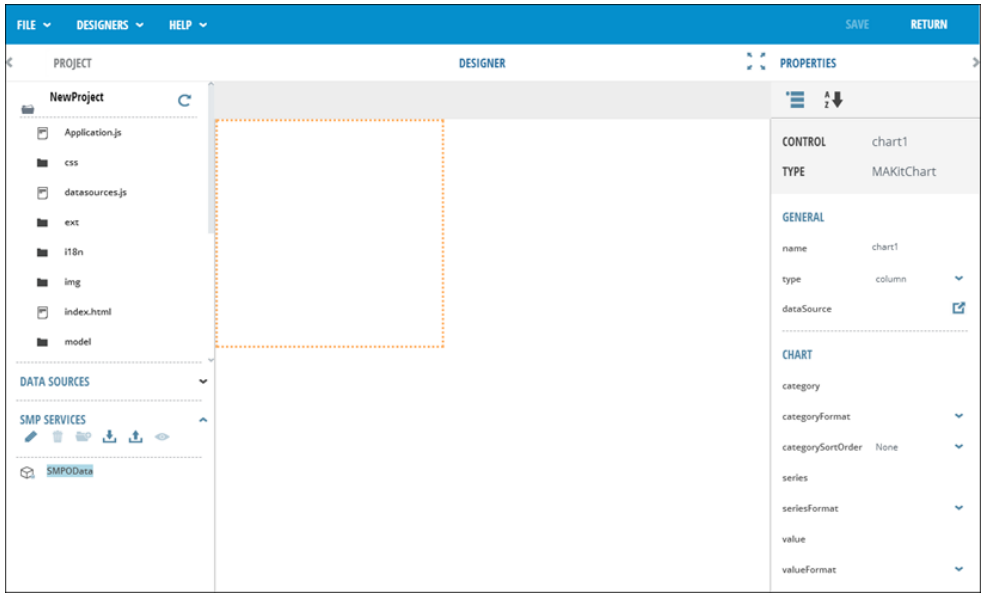


Chart Properties

Most chart properties are generic and similar to the other controls in the Form Designer. There are some properties that are specific to the chart control - `Type`, `dataSource`, and `Chart`. These properties define how to draw the chart and where to retrieve data:

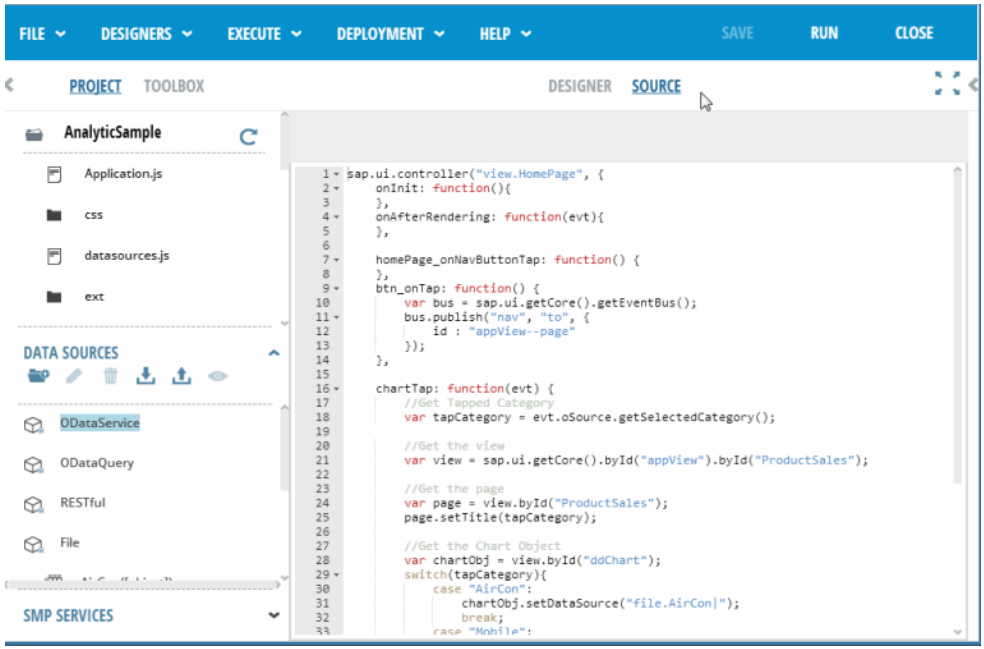
- `Type` – choose a type from the available list:
  - column
  - line
  - bubble
  - bar
  - pie
- `dataSource` – select the data source from which you will bind data to the control.
- `Chart` – configure the chart properties. These properties will vary based on the chart type chosen.

Source Code Editor

AppBuilder includes a Source Code Editor which you can use to customize code.

When you create a project, AppBuilder generates essential and skeleton code for the files that are autogenerated. While designing your project you can edit and customize this code using the Source Code Editor. The Source Code Editor is located in the center panel of the

AppBuilder work space, and you can toggle between the Form Designer and the Source Code Editor. Any changes that are made in the Form Designer are synchronized to the code when you switch to the source code page. You can modify the code in the editor, but be careful when you manually add code, as it can affect the structure of the designer page.



Developing Apps With AppBuilder

Follow the typical steps for developing generic or tab-based applications using AppBuilder.

1. *Creating a New Project*

   Use the New Project wizard to create a new project.
2. *Adding Controls to the Project*

   Add controls available in the Toolbox to your project.
3. *Configuring Controls in the Project*

   Use the Properties and Events panels to configure the appearance and behavior for your controls.
4. *Adding a New Page*

   Add a new page to your application.
5. *Testing the AppBuilder Project*

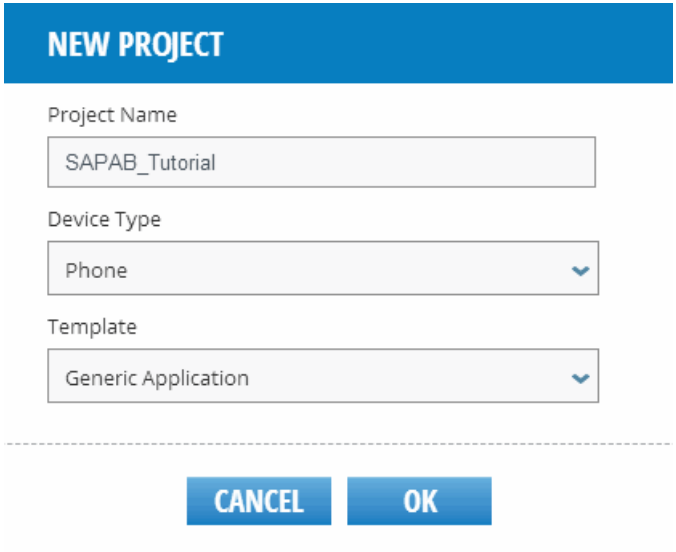
   Run the AppBuilder project in the browser to test that it is working properly.

Creating a New Project

Use the New Project wizard to create a new project.

1. In the AppBuilder Start Page, click **New**.
2. In the New Project wizard enter the following fields then click **OK**:
   a) Project Name – this will be the name assigned to the application. Do not include white spaces in your project name.
   b) Device Type – phone or tablet.
   c) Template – custom templates that you have created are listed under the provided Generic and Tab-Based Application templates.

      Tab-Based Application templates automatically place a tab at the bottom of the form on each page. When you run the application, all of the pages are displayed in the tab so that users can easily navigate between application pages.



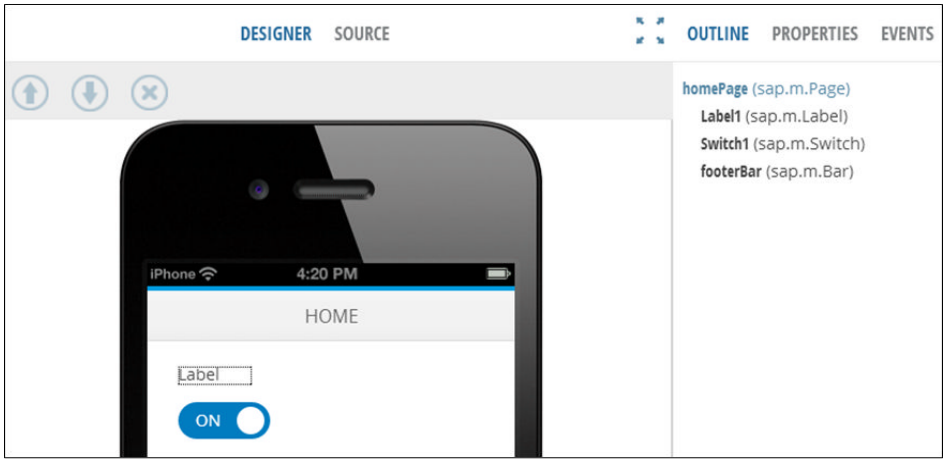
The project is created and opens in the Form Designer. Project files are generated and placed in `<AppBuilder_Home>\ares-project\hermes\filesystem\root\guest\<Project_Name>`.

Adding Controls to the Project

Add controls available in the Toolbox to your project.

The controls that are available for you to add will depend upon the Designer you are working with. The Toolbox is only available for use with the Form Designer and the SuperList Designer.

1. In your open project, select your control from the Toolbox.
2. Drag the control to the form in the Form Designer, and drop it in the form.



The control will appear graphically in the form, and also be listed in the Outline view for the page.

3. From the work space's main menu, click **Save**.

See also

- *Creating a New SuperList* on page 37
- *Creating a New Chart* on page 47
- *Configuring Controls in the Project* on page 25
- *Toolbox* on page 15

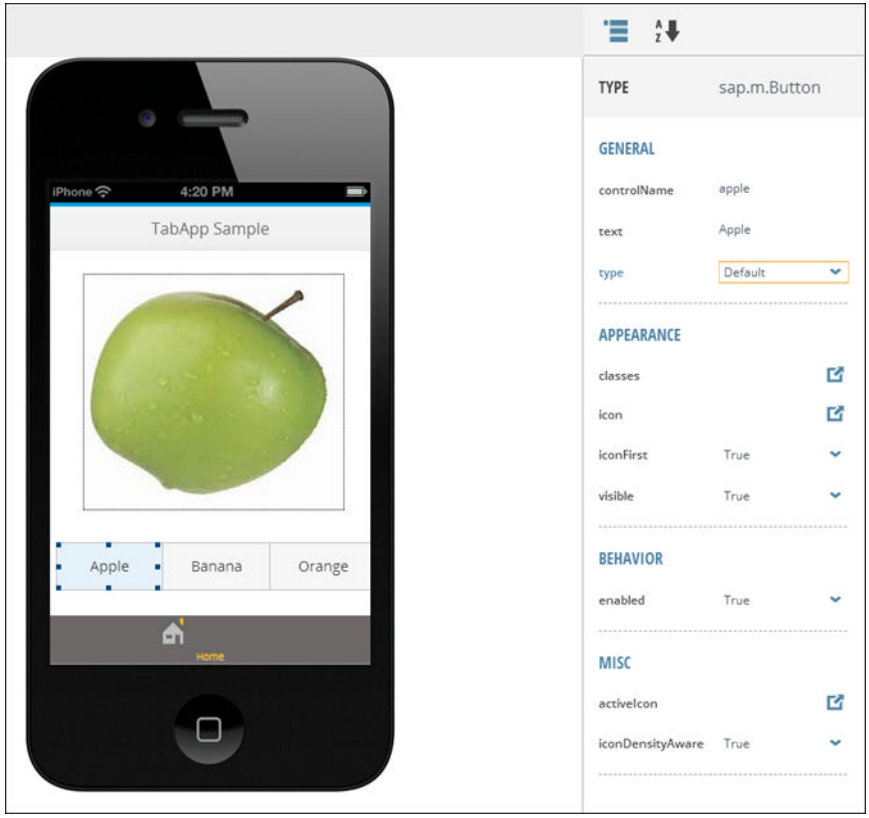
Configuring Controls in the Project

Use the Properties and Events panels to configure the appearance and behavior for your controls.

Each Designer in the AppBuilder workspace includes a properties panel that you can use to configure your properties. The Form Designer, also includes an Events panel that you can use to configure how the control will behave when interacted with. For example, you can specify the behavior of a button when the application user taps it.

Configuring Control Properties

In the form, click on your control to highlight it to display the available properties. You can also select it from the Outline panel.



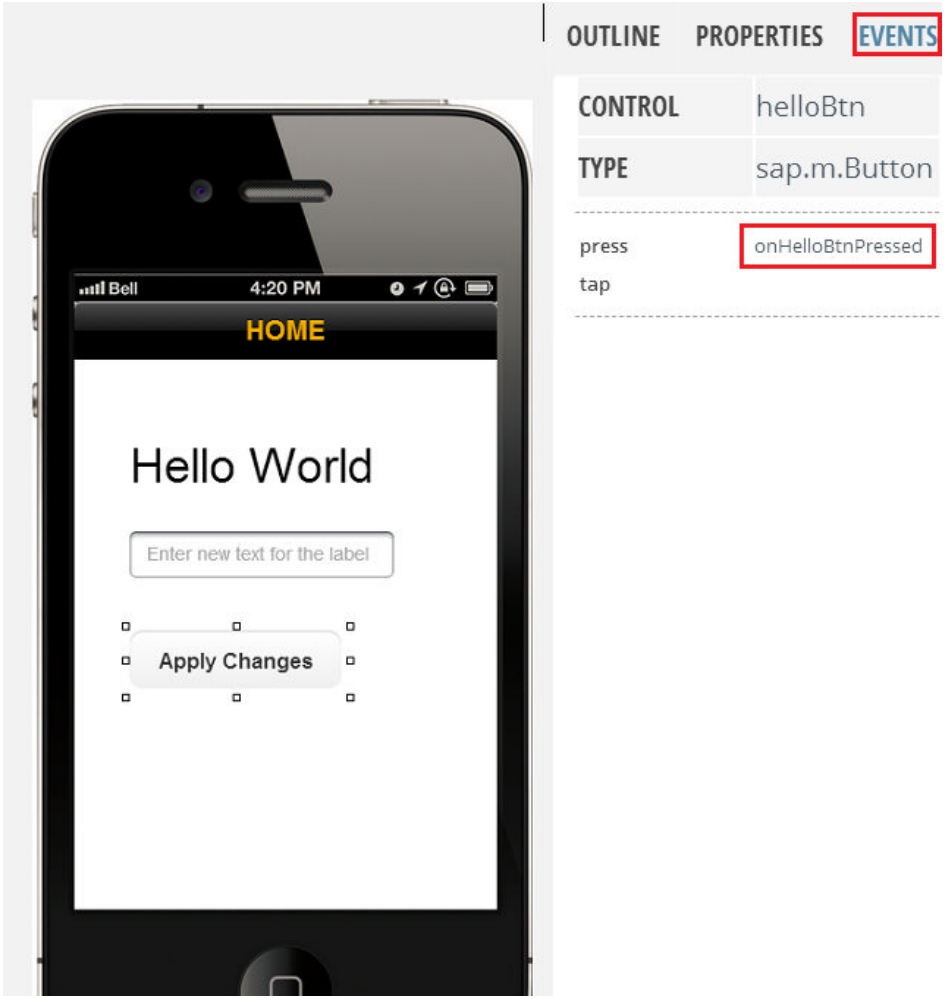
You can configure how the panel displays the properties by clicking the icons at the top of the panel. These icons allow you to toggle between an alphabetized listing of the properties, and the default categorized listing.

Most properties listed in the General category are simple naming and type properties. The properties listed in the Appearance category are used to assign image files, and associate style choices and cascading styles with your control. For example, when you click the classes property, the Classes Editor will appear, allowing you to add styles from the css folder in the Project Explorer. Each of the properties and categories in which they are grouped will vary based on the control selected.

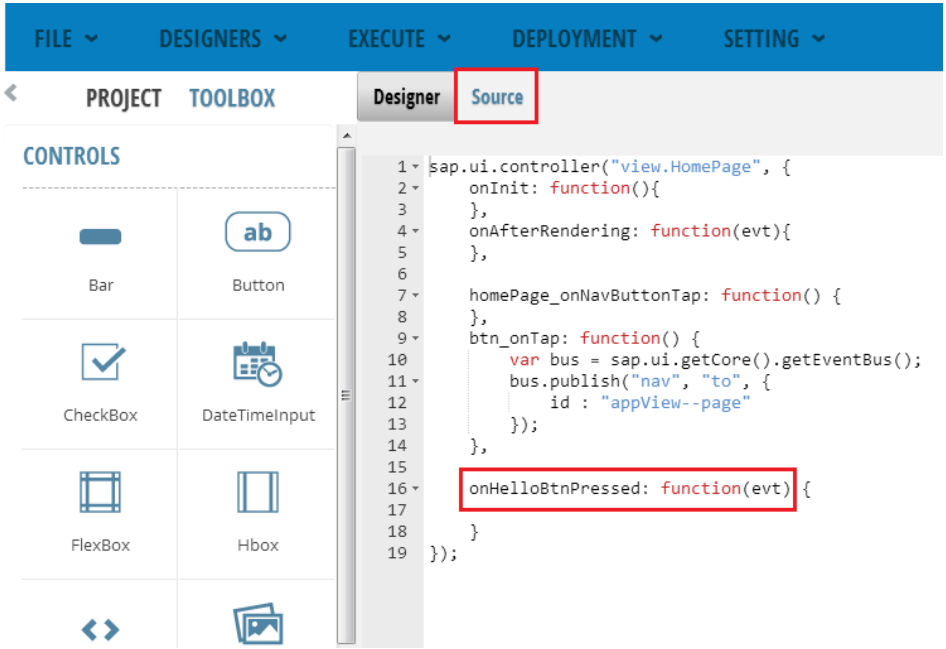
Configuring Control Events

When using the Form Designer, you can assign events to your controls to configure how they should behave when the user interacts with them. The example application in the image below is a simple application in which the user can enter new text in the blank field to modify the "Hello World" text that is displayed. To configure the behavior of the controls that create this event:

1. Select the control you wish to assign event to. In this example, it is the **Apply Changes** button



2. In the field next to the event you wish to assign, enter the event handler. For example, enter `onHelloBtnPressed`.
3. Click the **Source** tab to display the source code.

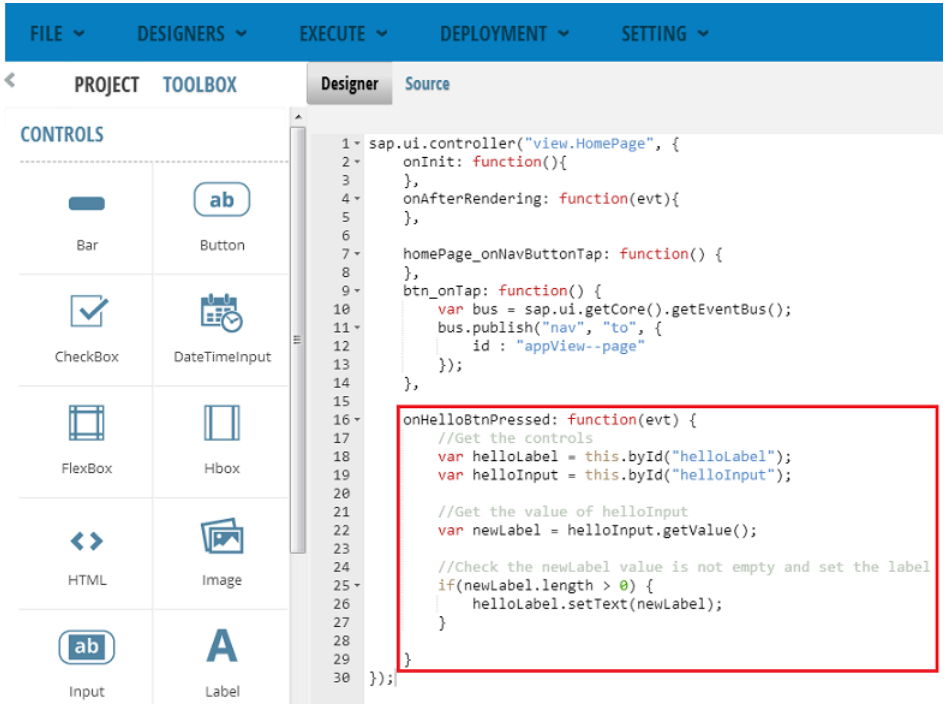


4. In the source code, enter the code to specify event handler actions. For example, enter the script below to specify behavior for the `onHelloBtnPressed` event handler.

```
//Get the controls
var helloLabel = this.byId("helloLabel");
var helloInput = this.byId("helloInput");

//Get the value of helloInput
var newLabel = helloInput.getValue();
//Check the newLavel value is not empty and set the label
if(newLabel.length > 0) {
    helloLabel.setText(newLabel);
}
```

The script is saved to the source code.



See also

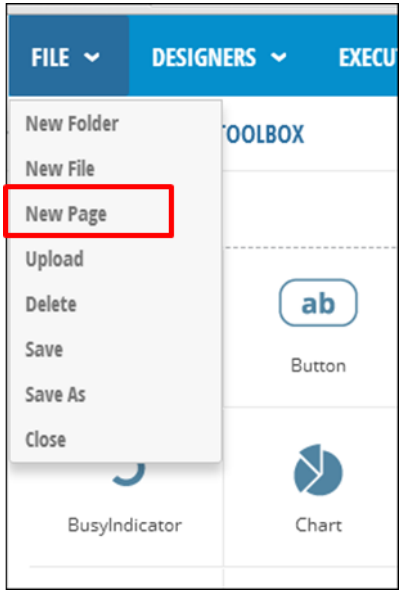
- *Toolbox* on page 15
- *Adding Controls to the Project* on page 24
- *Creating a New SuperList* on page 37
- *Creating a New Chart* on page 47

Adding a New Page

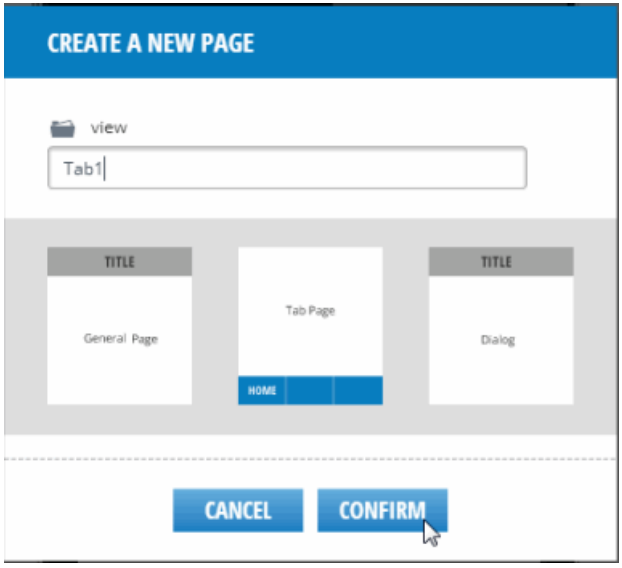
Add a new page to your application.

You can add a new page to either a generic application or a tab-based application using the File Menu. When working with tab based applications, adding a new page will automatically add tab navigation at the bottom of the form. When you run your tab-based application, all pages in the application are displayed as tabs at the bottom of the form. If you are developing a generic application, you will need to manually add and configure navigational controls between the pages. Generic applications are ideal if you don't want to expose the page linking on every page in your application.

1. In the main menu, choose **File** > **New Page**.



2. In the wizard, enter a name for the new page, choose the type of page, and click **Confirm**.
   - General page – a blank page with a title bar.
   - Tab page – a blank page with a segmented button control at tht bottom of the page.
   - Dialog page – a blank page with a title bar that has a configurable state property. The state property can be set to:
     - Error
     - Warning
     - Success
     - No state



Each tab page is created with `controller.js` and `view.html` files, located under the `view` folder in the Project view.

3. Add controls and event handlers to the tab pages as described in *Adding Controls to the Project* and *Configuring Controls in the Project*.
4. Run the project.

See also

- *Configuring Controls in the Project* on page 25
- *Toolbox* on page 15
- *Adding Controls to the Project* on page 24
- *Creating a New SuperList* on page 37
- *Creating a New Chart* on page 47

Testing the AppBuilder Project

Run the AppBuilder project in the browser to test that it is working properly.

After you have created your project, you can test it to ensure that your controls are properly configured by running the project in the browser.
In the AppBuilder menu click **Execute** > **Run**.

A new window opens in in your browser with your application displayed. From here, you can simulate interaction with it to test functionality.

Developing a SuperList App With AppBuilder

Follow the basic steps for developing an app containing a SuperList control using AppBuilder.

1. *Creating a New Project*

   Use the New Project wizard to create a new project.
2. *Creating a New Data Source*

   Use the data source panel to create a data object (or representative) for any OData service from the Web, or from a local JSON file.
3. *Creating a New SuperList*

   Use the SuperList Designer to create a new SuperList.
4. *Testing the AppBuilder Project*

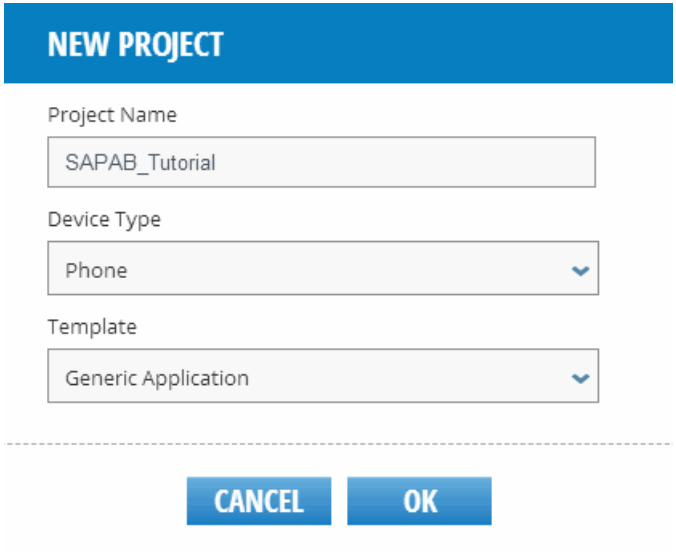
   Run the AppBuilder project in the browser to test that it is working properly.

Creating a New Project

Use the New Project wizard to create a new project.

1. In the AppBuilder Start Page, click **New**.
2. In the New Project wizard enter the following fields then click **OK**:
   a) Project Name – this will be the name assigned to the application. Do not include white spaces in your project name.
   b) Device Type – phone or tablet.
   c) Template – custom templates that you have created are listed under the provided Generic and Tab-Based Application templates.

      Tab-Based Application templates automatically place a tab at the bottom of the form on each page. When you run the application, all of the pages are displayed in the tab so that users can easily navigate between application pages.



The project is created and opens in the Form Designer. Project files are generated and placed in `<AppBuilder_Home>\ares-project\hermes\filesystem\root\guest\<Project_Name>`.

See also

- *Adding a Custom Theme* on page 60
- *Setting the Application Theme* on page 53
- *Project Management* on page 9
- *Saving a Project as a Template* on page 62

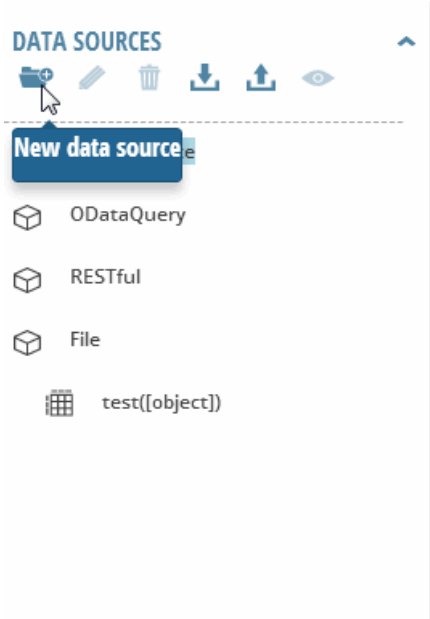
Creating a New Data Source

Use the data source panel to create a data object (or representative) for any OData service from the Web, or from a local JSON file.

Expand the data source panel to see all the defined data sources, with their field definitions.

Creating a New OData Data Source

1. In the Data Sources panel, click the new data source icon.



2. In the wizard, enter:
   - Name – enter a name for the data source
   - URL – enter the full URL for the data source
   - (Optional) User name – enter the user name with which to authenticate
   - (Optional) Password – enter the password with which to authenticate
   - Use JSONP – to use this option, the server to which you are making the cross domain request must be JSONP enabled.

     Note: The OData service must support JSONP. The SuperList update function cannot use this option.
   - Use Web server proxy – select this option to make calls to a Web server proxy instead of directly to the Web service. The proxy passes the call to the Web service and then passes the data back to the client application.

   If the OData service you are using does not reside on the same server and have the same domain as your developed Web app, the application cannot access the OData service because of the "same origin" policy. See *Same Origin* policy. In these cases, you can choose either the "Use JSONP" option or the "Use Web server proxy" option.
3. Click **Confirm**.

Creating a Data Source Using a Local JSON File

The JSON file you use must be in the JSON format. For example, if the data is in an array, it should look like this:

```
[
    { "id": 1, "name": "Smith", "Salary": 5000.00, "Gender": true },
    { "id": 2, "name": "John", "Salary": 4000.00, "Gender": true },
```

```
    { "id": 3, "name": "Lilian", "Salary": 7000.00, "Gender": false }
[
```

The JSON file must be located under the IDE Root Folder, `ares-project`, or one of its sub folders. For the URL, enter either the full URL or the IP address for the JSON file.

For example, if your JSON file is located in `ares-project/ares/employee.json`, define the URL for the data object as:

`http://127.0.0.1:9009//ide/ares/employee.json`, or as `/ide/ares/employee.json`.

Using a Remote OData Service With no JSONP Support

Select the **Use Web Server Proxy** option.

Using a Remote OData Service With Parameters

You can use a remote OData service with parameters in situations where you have many OData services that use the same root and most of the same URL, with only minor differences between them, for example:

- http://www.mycompany.com/corp/department
- http://www.mycompany.com/corp/employee
- http://www.mycompany.com/corp/finance

In this example, three data objects were created normally, and the objects can be called individually. To call the OData service dynamically, you can create a single generic data object with parameters defined in the URL. For example:

http://www.mycompany.com/corp/{?}

{?} is defined to represent one parameter, so that when the OData service is called, only one parameter is passed and the {?} is replaced with that parameter. ? is an integer starting with 0.

See also

- *Creating a New SuperList* on page 37

Importing a Data Source

Import a data source.

You can import a data source in the data source panel.

1. In the data sources panel, click the import data sources icon.

2. Browse to the location of the dsconfig file you wish to import. The file is uploaded to your server via HTTP, and added to the datasource list in the data sources panel. The configuration for the datasource is also added to the datasources.js file in the

`<AppBuilder_Home>\ares-project\hermes\filesystem\root\guest\<Project_Name>\src` directory.

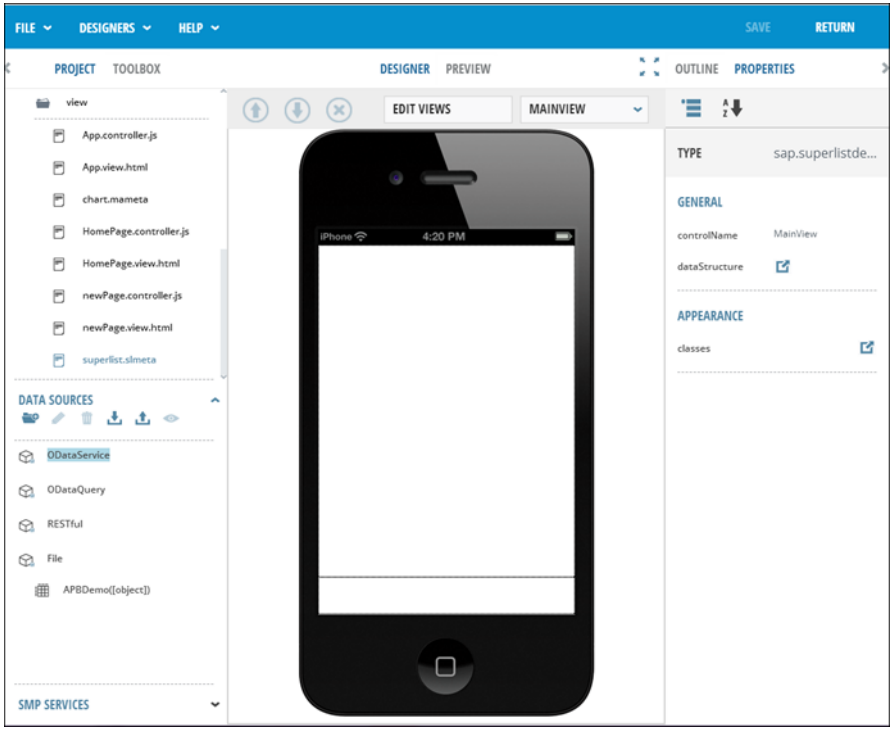
Creating a New SuperList

Use the SuperList Designer to create a new SuperList.

A SuperList is a control that acts as a container for controls that are bound to data from a datasource. To create the SuperList, create a metadata file and add it to the SuperList control following these steps:

1. From the menu, select **Designers** > **New SuperList**.
2. In the dialog, enter the information and click **Confirm**:
   - File name – name for the SuperList, ending with the .slmeta extension.
   - View Name – name for the view
   - View style – choose from:
     - ListView – displays data from from your specified datasource in list format on your device screen.
     - FormView – displays a row of data from your specified datasource on your device.
     - GroupView – displays data from your datasource grouped by the column that you specify.

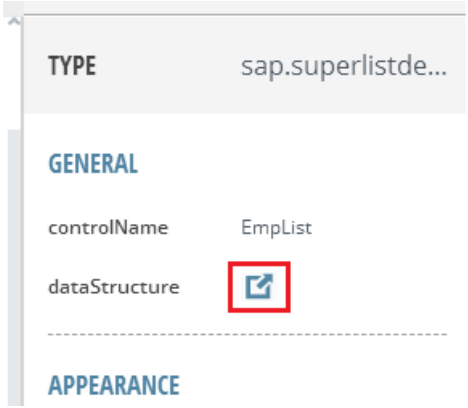
   The new view opens in the SuperList designer as shown.



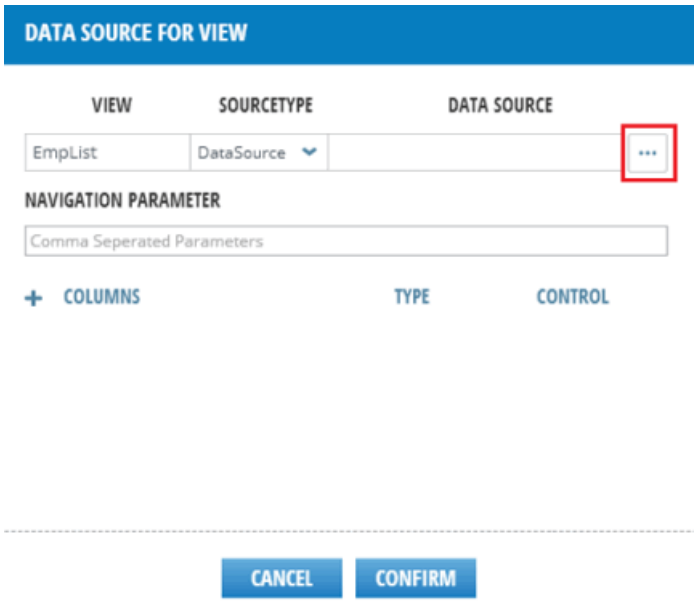
3. (Optional) Add another view to the SuperList:
   a) Click **Edit Views**.
   b) Click **Add**.
   c) Enter a name for the new view, select the view type, and click **Confirm**.

      If there is more than one view available for the SuperList, you can select a view from the Views List dropdown next to the **Edit Views** button.

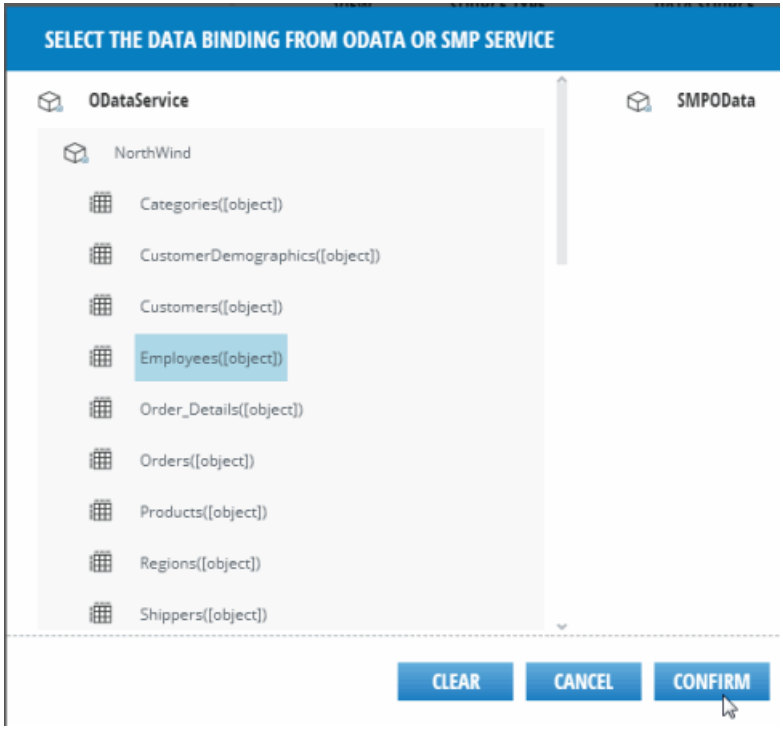
      When editing views, you can preview the view result by clicking **Preview**.
4. Bind data to the control:
   a) Go to Properties panel and click the dataStructure icon.



b) Click Data Source.



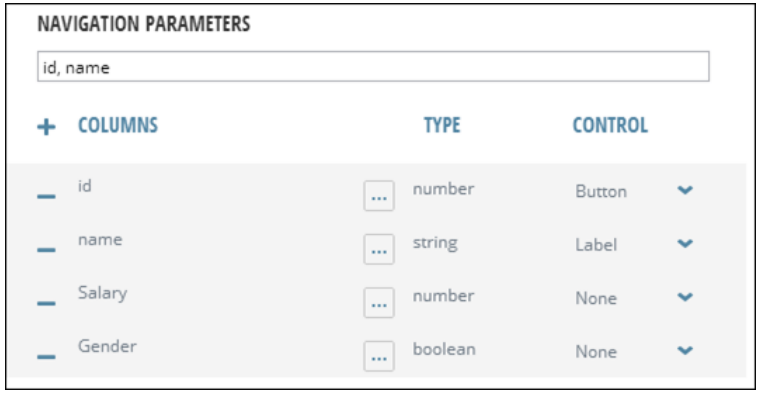
c) Select the data binding, and click Confirm.



The Data Source for View window appears and lists available columns.

d) In the Data Source for View window, configure the columns that you want to display in your superlist and assign controls:

1. In the Navigation Parameters field, enter comma separated column names that you wish to pass to the next level list as drill-down navigation parameters.



You can also double-click on each column name that you would like to specify as a navigation parameter to add them in the Navigation Parameters field.

2. (Optional) Click the

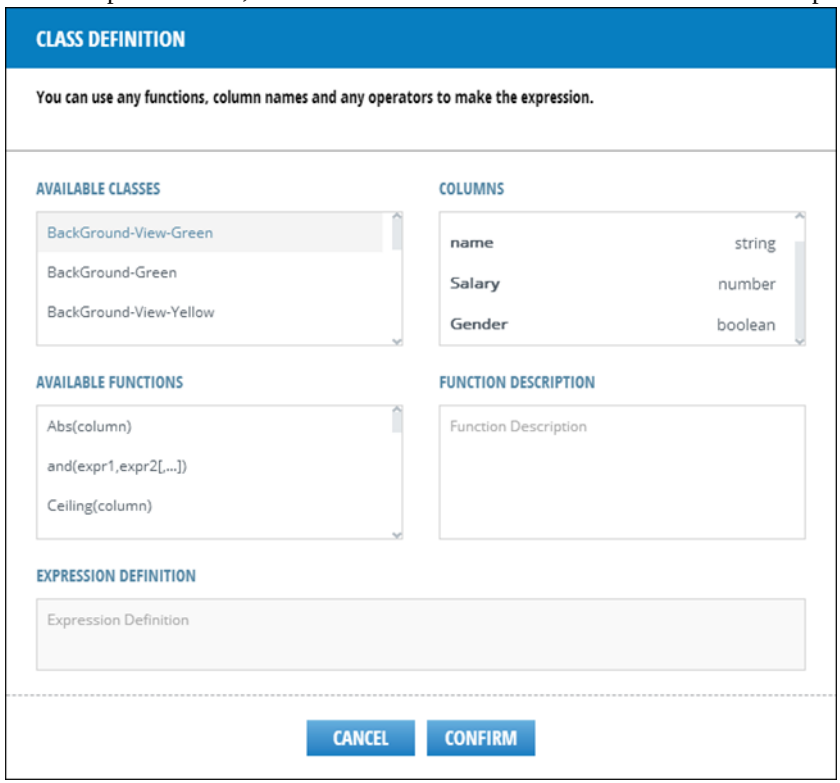
icon to modify the columns.

3. (Optional) Click the drop-down arrow in the Control column to display the list of control types, and select one to bind to the specified column.
4. Click **Confirm** to save your choices. The controls and associated column names are added to the form in the Designer.

e) (Optional) While working in the SuperList Designer, you can add and configure a limited number of controls. Configure the display and behavior for the data and the controls for the columns you wish to include in your SuperList.

You can add controls to your SuperList from the Form Designer or from the SuperList Designer. However, not all of the controls in the Toolbox palette are available from the SuperList Designer. See *ToolBox* for a list of the controls that are available while working in the SuperList Designer.

1. In the form, select the control you wish to configure.
2. In the Properties Panel, click the classes icon. The Class Definition window opens.



3. Use any functions, column names, and any operators from the lists provided to create your expression in the Expression Definition field.

4. Click Confirm to save your choices. Repeat these steps for each control you wish to configure.

f) Select **Return** on the AppBuilder menu to exit the SuperList Designer.

5. Click **Toolbox**.
6. Drag and drop the SuperList control onto the form.
7. In Properties, click the metaData File icon.
8. Under the view folder, select the `<file_name>.slmeta` file and click **Confirm**.

   The data appears in the control on the form. After the metadata file is set, you can reconfigure the datasource in the Properties panel as needed.

See also

- *Creating a New Data Source* on page 34
- *Adding Controls to the Project* on page 24
- *Creating a New Chart* on page 47
- *Configuring Controls in the Project* on page 25
- *Toolbox* on page 15
- *SuperList Designer* on page 19

Testing the AppBuilder Project

Run the AppBuilder project in the browser to test that it is working properly.

After you have created your project, you can test it to ensure that your controls are properly configured by running the project in the browser.
In the AppBuilder menu click **Execute** > **Run**.

A new window opens in in your browser with your application displayed. From here, you can simulate interaction with it to test functionality.

Developing a Chart App With AppBuilder

Follow the basic steps for developing an app containing a Chart control using AppBuilder.

1. *Creating a New Project*

   Use the New Project wizard to create a new project.
2. *Creating a New Data Source*

   Use the data source panel to create a data object (or representative) for any OData service from the Web, or from a local JSON file.
3. *Creating a New Chart*

   Use the Chart Designer to create a new chart
4. *Testing the AppBuilder Project*

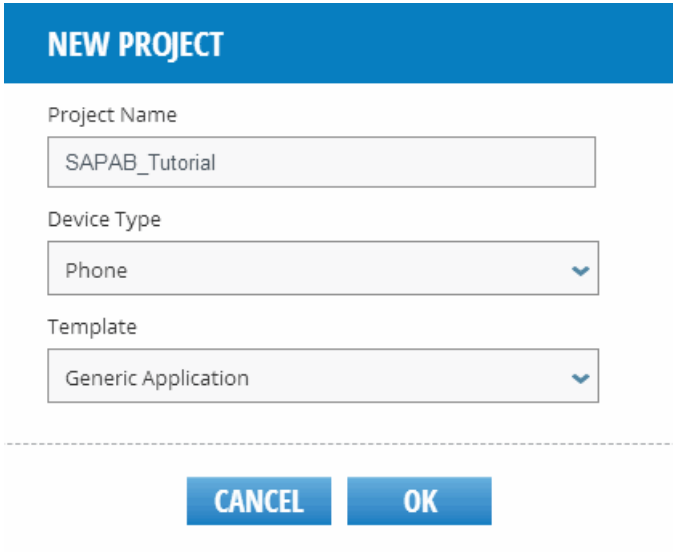
   Run the AppBuilder project in the browser to test that it is working properly.

Creating a New Project

Use the New Project wizard to create a new project.

1. In the AppBuilder Start Page, click **New**.
2. In the New Project wizard enter the following fields then click **OK**:
   a) Project Name – this will be the name assigned to the application. Do not include white spaces in your project name.
   b) Device Type – phone or tablet.
   c) Template – custom templates that you have created are listed under the provided Generic and Tab-Based Application templates.

      Tab-Based Application templates automatically place a tab at the bottom of the form on each page. When you run the application, all of the pages are displayed in the tab so that users can easily navigate between application pages.



The project is created and opens in the Form Designer. Project files are generated and placed in `<AppBuilder_Home>\ares-project\hermes\filesystem\root\guest\<Project_Name>`.

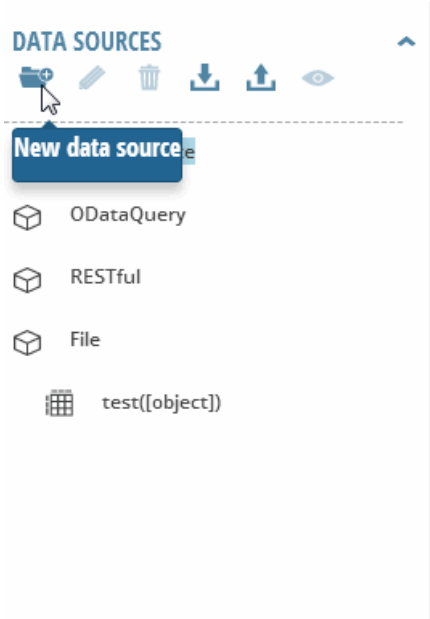
Creating a New Data Source

Use the data source panel to create a data object (or representative) for any OData service from the Web, or from a local JSON file.

Expand the data source panel to see all the defined data sources, with their field definitions.

Creating a New OData Data Source

1. In the Data Sources panel, click the new data source icon.



2. In the wizard, enter:
   - Name – enter a name for the data source
   - URL – enter the full URL for the data source
   - (Optional) User name – enter the user name with which to authenticate
   - (Optional) Password – enter the password with which to authenticate
   - Use JSONP – to use this option, the server to which you are making the cross domain request must be JSONP enabled.

     Note: The OData service must support JSONP. The SuperList update function cannot use this option.
   - Use Web server proxy – select this option to make calls to a Web server proxy instead of directly to the Web service. The proxy passes the call to the Web service and then passes the data back to the client application.

   If the OData service you are using does not reside on the same server and have the same domain as your developed Web app, the application cannot access the OData service because of the "same origin" policy. See *Same Origin* policy. In these cases, you can choose either the "Use JSONP" option or the "Use Web server proxy" option.
3. Click **Confirm**.

Creating a Data Source Using a Local JSON File

The JSON file you use must be in the JSON format. For example, if the data is in an array, it should look like this:

```
[
    { "id": 1, "name": "Smith", "Salary": 5000.00, "Gender": true },
    { "id": 2, "name": "John", "Salary": 4000.00, "Gender": true },
```

```
    { "id": 3, "name": "Lilian", "Salary": 7000.00, "Gender": false }
[
```

The JSON file must be located under the IDE Root Folder, `ares-project`, or one of its sub folders. For the URL, enter either the full URL or the IP address for the JSON file.

For example, if your JSON file is located in `ares-project/ares/employee.json`, define the URL for the data object as:

`http://127.0.0.1:9009//ide/ares/employee.json`, or as `/ide/ares/employee.json`.

Using a Remote OData Service With no JSONP Support

Select the **Use Web Server Proxy** option.

Using a Remote OData Service With Parameters

You can use a remote OData service with parameters in situations where you have many OData services that use the same root and most of the same URL, with only minor differences between them, for example:

- http://www.mycompany.com/corp/department
- http://www.mycompany.com/corp/employee
- http://www.mycompany.com/corp/finance

In this example, three data objects were created normally, and the objects can be called individually. To call the OData service dynamically, you can create a single generic data object with parameters defined in the URL. For example:

http://www.mycompany.com/corp/{?}

{?} is defined to represent one parameter, so that when the OData service is called, only one parameter is passed and the {?} is replaced with that parameter. ? is an integer starting with 0.

See also

- *Creating a New Chart* on page 47

Importing a Data Source

Import a data source.

You can import a data source in the data source panel.

1. In the data sources panel, click the import data sources icon.

2. Browse to the location of the dsconfig file you wish to import. The file is uploaded to your server via HTTP, and added to the datasource list in the data sources panel. The configuration for the datasource is also added to the datasources.js file in the

`<AppBuilder_Home>\ares-project\hermes\filesystem\root\guest\<Project_Name>\src` directory.

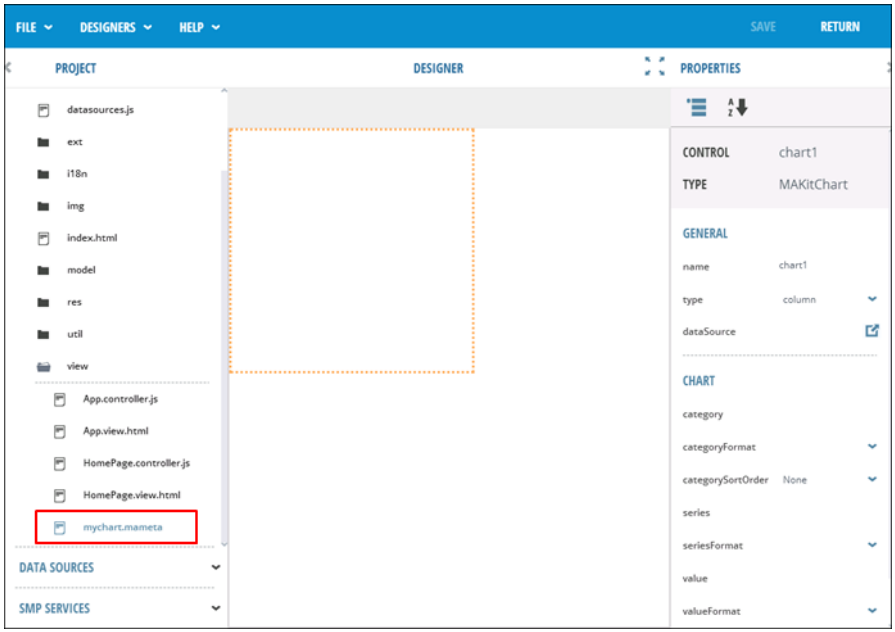
Creating a New Chart

Use the Chart Designer to create a new chart

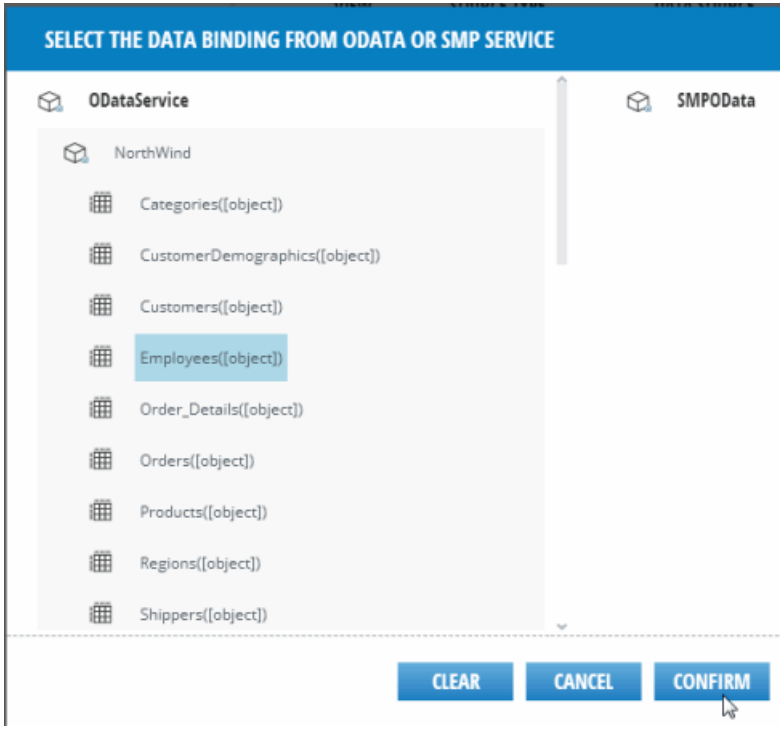
A chart is a control that graphically displays data from a datasource. To create the chart, create a metadata file and add it to the chart control following these steps:

1. In the main menu of the Form Designer, select **Designers > New Chart**.
2. In the wizard, enter a name for the metadata, for example, and click **Confirm**.

   The metadata file is created under the `view` folder in the Project Explorer.



3. Bind data to the control.
   a) Go to the Properties panel and select the dataSource icon.
   b) In the data binding catalog, select the data source, and click Confirm.

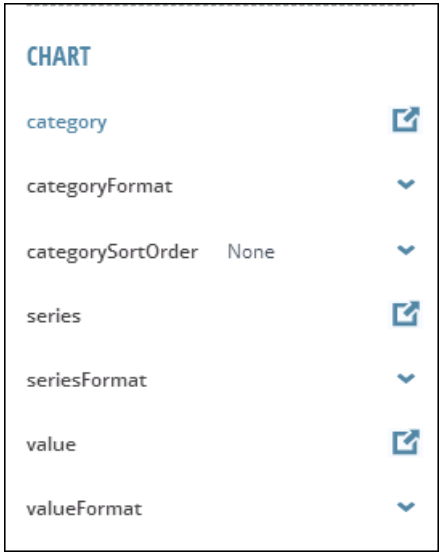


The Data Source is added to the dataSource property for the chart.

4. Configure the chart control.

a) In the Properties panel, click **type** to select the chart type, for example Column.

b) In the Properties panel, specify the control properties for the chart.



Note: Important notes for configuring chart properties:

- The category and value properties are required properties that must be set. The series property is optional.
- Sorting – AppBuilder's charting feature requires its data to be sorted by category column. If you set the categorySortOrder to **None**, then you need to make sure the date is pre-sorted by category. It doesn't matter how it is sorted (for example, alphabetically), as long as it is sorted.
- Duplicated Data – AppBuilder's charting feature does not automatically aggregate data, and it expects that each row of data is unique based on category and series (if the series property is set). If there are rows in the data with the same category and series values, the duplicate row is removed, and the data is only displayed once on the Chart.

5. Click **Save** to save your chart, and then click **Return** to exit the Chart Designer.
6. Click **Toolbox**.
7. Drag and drop the Chart control onto the form.
8. In Properties, click the metadataFile icon.
9. Under the view folder, select the `<file_name>.mameta` file and click **Confirm**.

   The chart appears on the form.
10. Click **Save**.

See also

- *Creating a New Data Source* on page 44
- *Adding Controls to the Project* on page 24
- *Creating a New SuperList* on page 37

- *Configuring Controls in the Project* on page 25
- *Toolbox* on page 15
- *Chart Designer* on page 19

Testing the AppBuilder Project

Run the AppBuilder project in the browser to test that it is working properly.

After you have created your project, you can test it to ensure that your controls are properly configured by running the project in the browser.
In the AppBuilder menu click **Execute** > **Run**.

A new window opens in in your browser with your application displayed. From here, you can simulate interaction with it to test functionality.

Configuring a Project

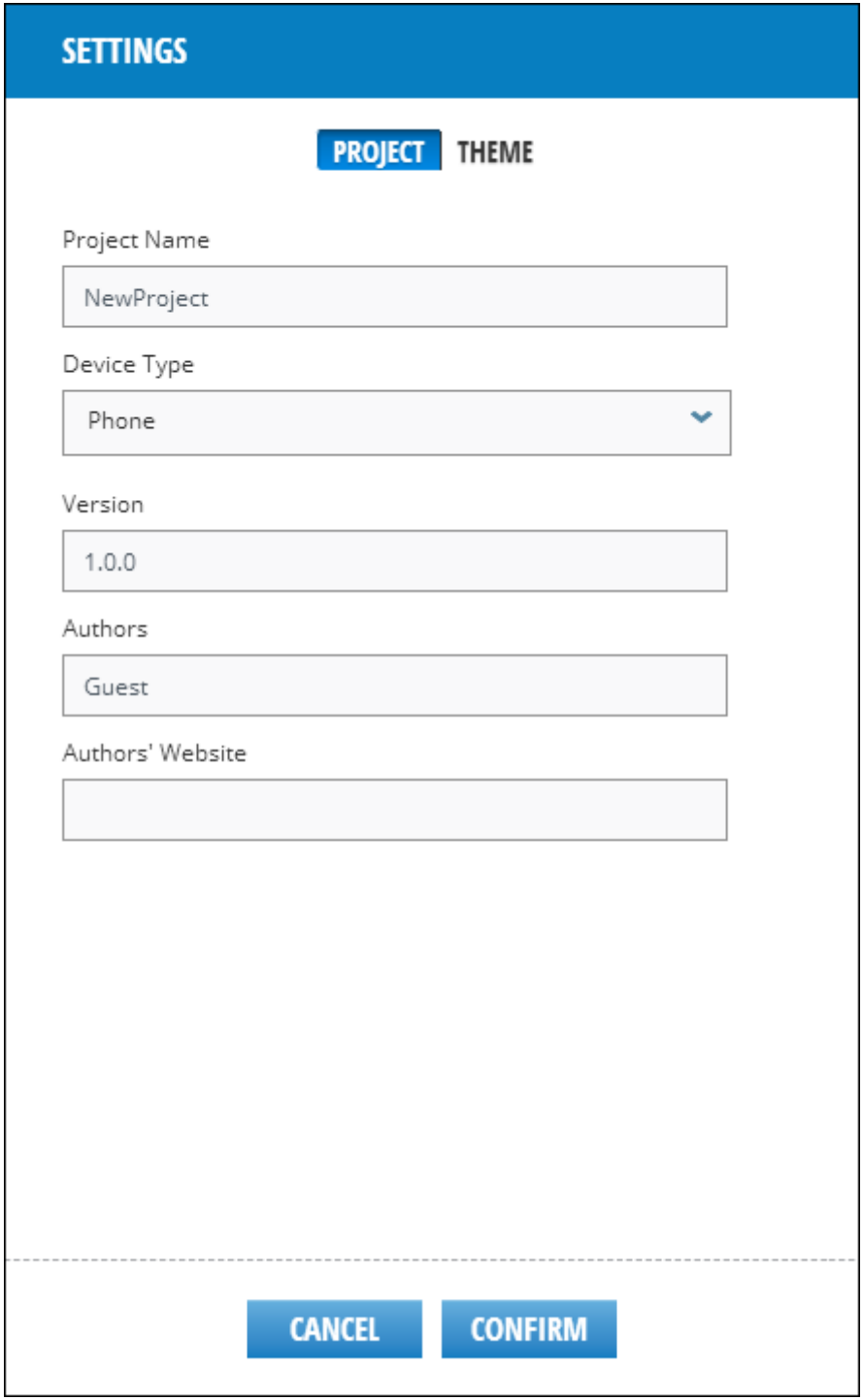
AppBuilder provides several options for configuring AppBuilder projects.

You can configure the basic settings and modify the themes and styles for your project. You can also save your project as a template for use in creating new projects.

Configuring Project Settings

Configure project settings such as device type, name, version, and so on.

1. From the start page, select the project you wish to configure and click **Settings**.
2. In the Settings wizard, configure the desired project settings by modifying one or more of the fields listed, then click **Confirm**:
   - Project Name
   - Device Type
   - Version
   - Authors
   - Author's Website


SETTINGS
PROJECT
THEME
Project Name
NewProject
Device Type
Phone
Version
1.0.0
Authors
Guest
Authors' Website
CANCEL
CONFIRM

Themes and Styles

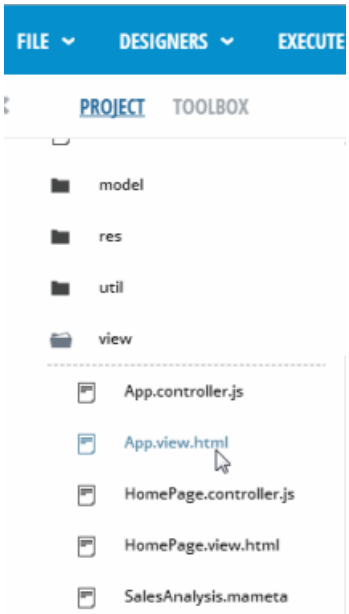
Two predefined themes are provided with AppBuilder—BlueCrystal (default) and MVI.

You can also add your own customized themes.

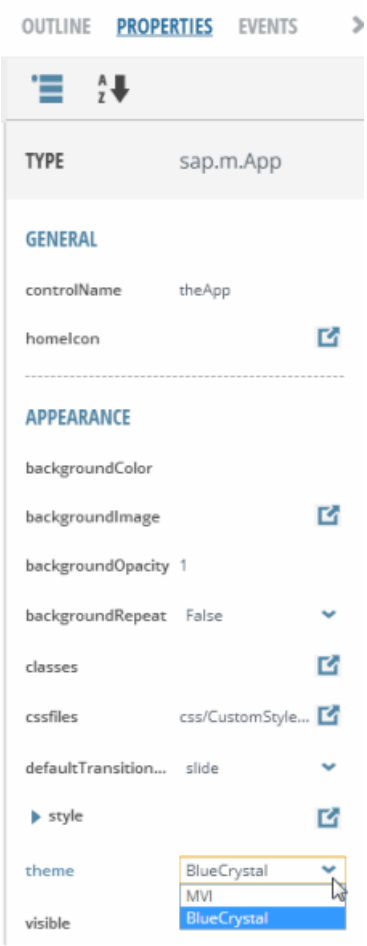
Setting the Application Theme

Set the theme for the application in the Properties view.

1. In the form designer, select the application for which you want to change the theme.
2. In the Project browser, double-click the `App.view.html` file.



3. In **Properties** > **Appearance** > **Theme**, choose the theme.



4. Click **Save**.

See also

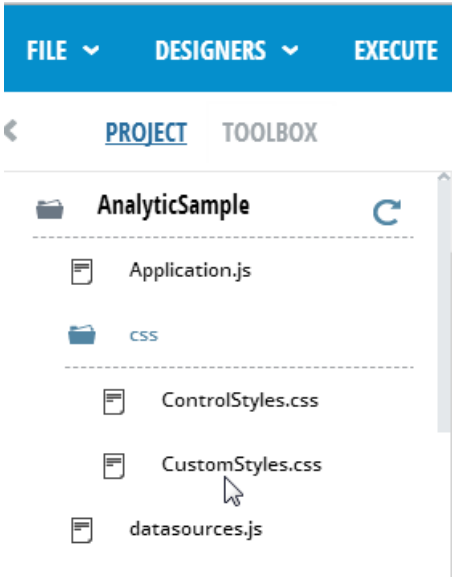
- *Adding a Custom Theme* on page 60
- *Creating a New Project* on page 33
- *Project Management* on page 9
- *Saving a Project as a Template* on page 62

Customizing Styles

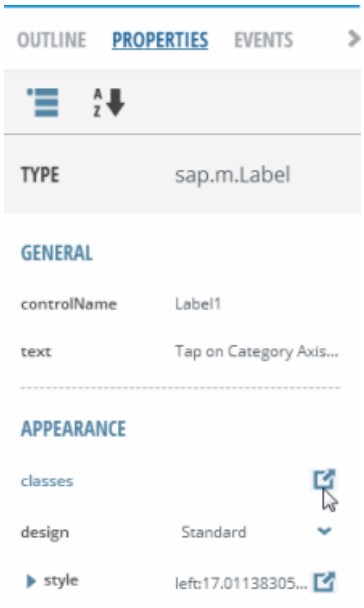
Add custom classes to the `CustomStyles.css` file to create customized styles in the theme definition.

The `CustomStyles.css` file is located in the application's `css` folder.

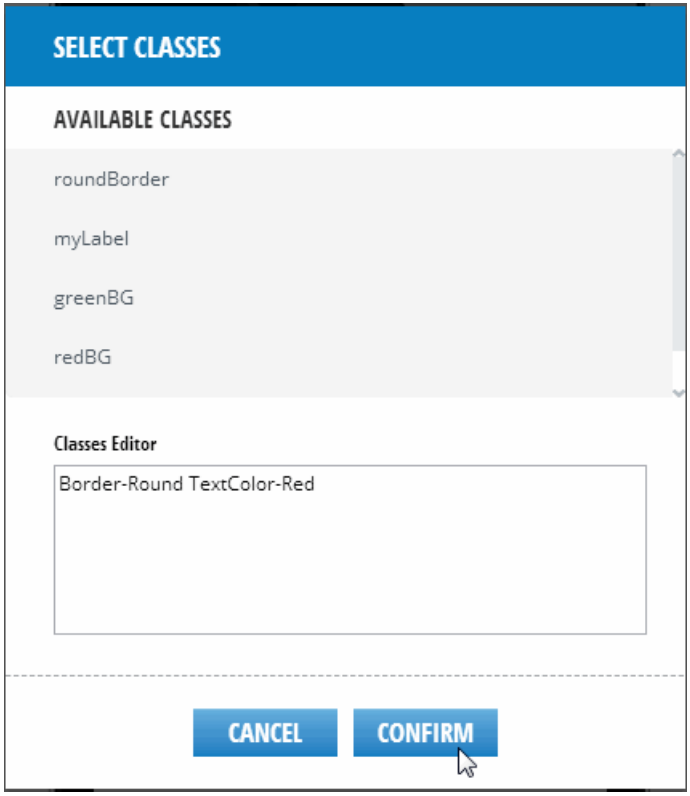
1. In the Project browser, double-click the `CustomStyles.css` file.



2. Add your custom classes to the source code, and click **Save**.
3. In **Properties** > **Appearance**, click the classes icon.



4. In the dialog, in the Classes Editor, add your custom class, and click **Confirm**.



5. Click **Save**.

Example Styles

AppBuilder uses classes and elements in `customstyles.css`.

The priority value is (a,b,c,d), a=0,b=0. For the style to take effect, do not make the style's priority lower than the default UI5 style. The priority level of the selector is decided in Point of combination of selectors. The priority is calculated by (a,b,c,d).

- Style attribute = a
- Number of ID attributes = b
- Number of classes = c
- Number of element names = d

For example, if there is a combination of the following selectors in `index.html`:

```
<body>
  <article>
    <p>This is <span id="red">paragraph</span>.</p>
  </article>
</body>
```

And this combination of selectors in `customstyles.css`:

```
article p span{
   color: blue;
 }
 #red{
   color: red;
 }
```

The paragraph becomes a red character because `#red(0100)` is bigger than `article p span(0003)`.

The !important Rule

There are some circumstances when you may want to apply a specific style for an element but the same style for that element has also been declared elsewhere in the style sheet or in the document.

For example, there is an anchor tag with embedded styles:

```
<a style="color: #border: 1px solid #156E8E; background-color: #389ABE;">This is a link</a>
```

And there is also a separate style for that anchor tag in the style sheet:

```
a {
    border: 1px solid #333;
    background-color: #555;
}
```

In this example, you can use the `!important` rule to force the browser to use the styles in the style sheet instead of the one in the element:

```
a {
border: 1px solid #333 !important;
background-color: #555 !important;
}
```

The `!important` rule indicates that this style is the most important and must be applied over the other styles, even when it is directly embedded in the element.

Button Background

The default button background style is defined in the `library.css` file.

```
.sap-ios .sapMBtnDefault{
              linear-gradient(top,#ffffff 0,#f0f0f0 100%)
}
```

The priority value is (0,0,2,0), so to make new added classes take effect, make the new style priority no lower than the default. The following is defined in `customstyles.css` and its priority is (0,0,2,0).

```
.sap-ios .greenBG {
    background: -webkit-gradient(linear, left top, left bottom,
color-stop(0%,#cdeb8e), color-stop(100%,#a5c956));
}
```

If the priority is same, the style that is loaded later takes higher priority. In this example, the `customstyles.css` is loaded later than library.css, so what is defined in the `customstyles.css` take priority.

The Bar Background

The bar control's default style is defined in library.css and has the priority value (0,0,2,1):

```
.sap-ios .sapMBar:not(.sapMBarTranslucent) {
webkit-linear-gradient(top, rgb(140, 140, 140) 0px, rgb(64, 64, 64)
100%);
}
```

`.sap-ios .greenBG` will not take effect, because its priority is (0,0,2,0). You can change it to have the priority (0,0,3,0):

```
.sap-ios .sapUiView .greenBG{
webkit-linear-gradient(top, rgb(205, 235, 142) 0%, rgb(165, 201, 86)
100%);
}
```

The !important Rule

If you want to apply the new styles and do not want to touch details of the class's structure, you can use the !important rule, for example:

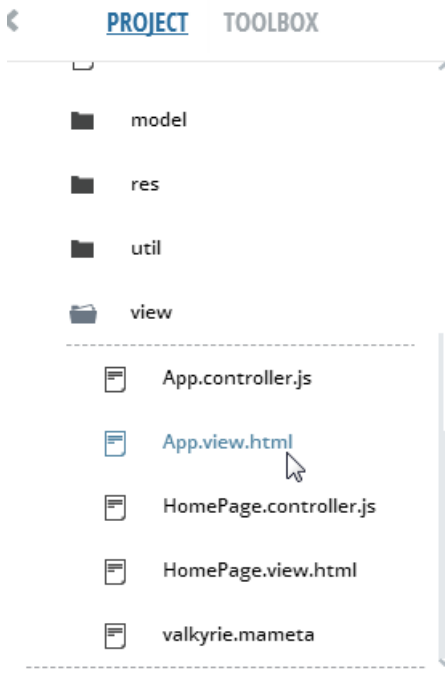
```
.greenBG{
webkit-linear-gradient(top, rgb(205, 235, 142) 0%, rgb(165, 201, 86)
100%) !important;
}
```

Adding Custom Styles Files

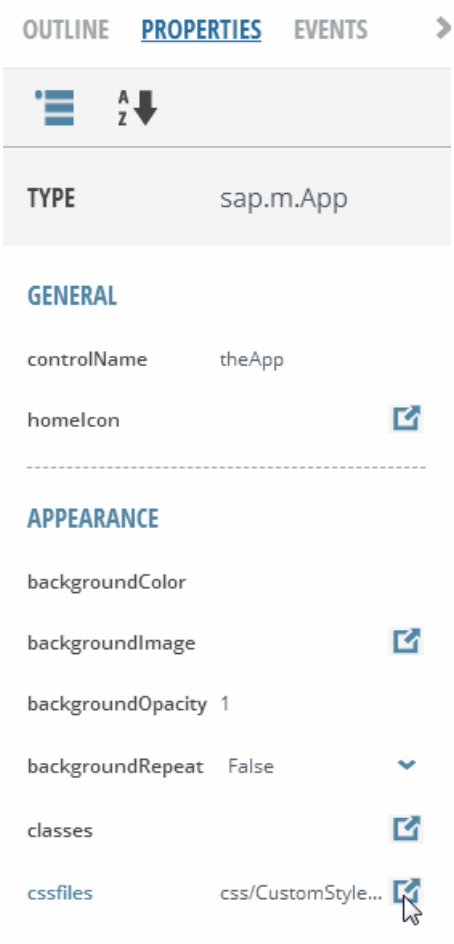
You can add other CSS files as custom style files whose classes will be listed in the Select Classes dialog.

By default, the custom style file is `CSS/CustomStyles.css`, which is in application's folder and relative to its `index.html` file.

1. In the Project files view, double-click **App.view.html** to open the CSS properties.



2. In Properties, click the icon next to cssfiles.



3. In the Custom CSS Files dialog, navigate to your custom css file, and click **Add**.

Adding a Custom Theme

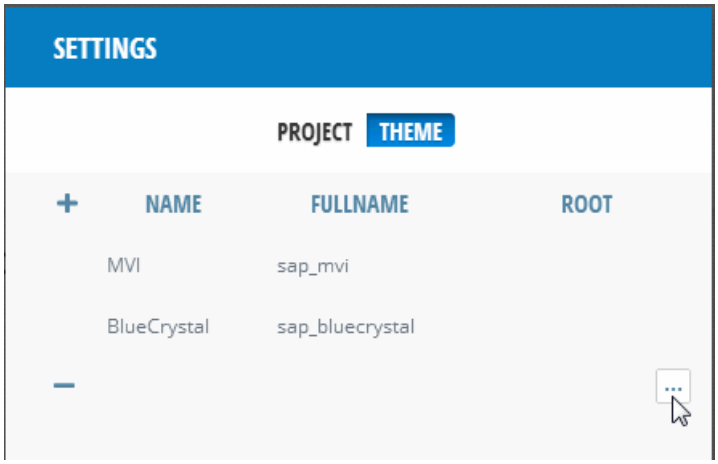
Add a custom theme to apply to your applications.

AppBuilder provides two default themes—MVI and BlueCrystal. These themes cannot be edited, but you can customize your own them and add it to the library of themes. You can then apply the theme to your application as described in *Setting the Application Theme*. The default location for themes in the AppBuilder directory structure is `<AppBuilder-Home>\ares-project\ares\ui5\resources`.

1. Add your theme folder to the default AppBuilder themes directory, or to the file structure for your project. For example, to apply a custom theme to a project named "My Project",

add the folder to `<AppBuilder-Home>\ares-project\hermes\root\guest\<My_Project>\src\css\`.

2. In the AppBuilder start page, select the application for which to add the theme, and click **Settings**.
3. In the Settings dialog, click **Themes** to display the theme settings.



The default settings available to the application appear in the theme list.

4. Click + to add a row for the custom theme settings.
5. Enter the settings for the new theme:
   - Name - the display name which will be displayed in the application's theme property.
   - Fullname - the internal name which used for the `applyTheme` API.
   - Root - the location where you placed the theme folder in Step 1. Leave blank if you placed the theme folder in the default AppBuilder `themes` directory. Default themes are in the `ui5\resources` directory. For new themes, refer to the theme folder structure in `ui5\resources`.
6. Click **Confirm** to save your settings.
   When you complete the steps in *Setting the Application Theme*, the new theme that you added is available in the application theme property list.

See also

- *Setting the Application Theme* on page 53
- *Creating a New Project* on page 33
- *Project Management* on page 9
- *Saving a Project as a Template* on page 62

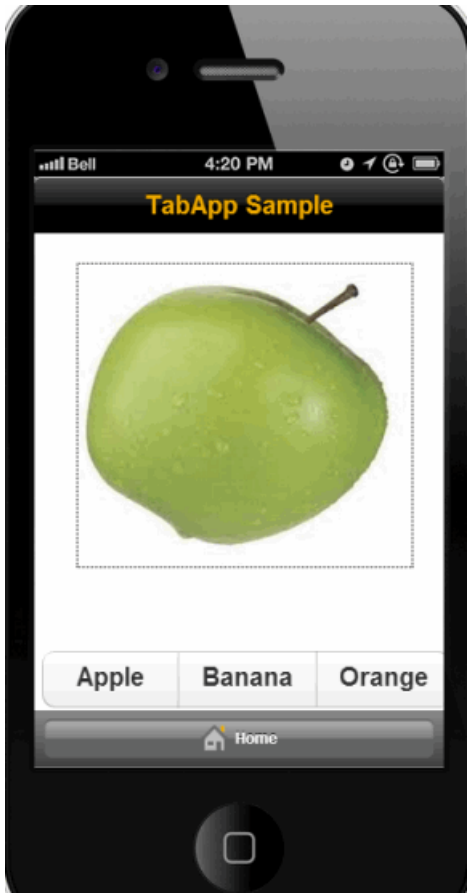
Saving a Project as a Template

Save an existing project as a template.

When you create a new project, you can select one of two different types of application templates for your project. When you save a project as a template, you will also generate one these template types to expedite application development for similar applications. The two basic template types available in AppBuilder are:

Generic Application Template
This template creates a basic application.

Tab Based Application Template
Tab based applications display a series of buttons along the bottom of the screen. Tapping the different buttons in the application presents a different screen of data, or, a new root view.



Saving a project as a template

You can also save an existing project to create a custom template for future reuse:

1. From the start page, select the project you wish to create the project from, and click **Template**.
2. Enter a name for your template (if different from the project name).
3. Click **Save**. The template will be saved to your list of templates and be made available when you create new projects.

See also

- *Project Management* on page 9
- *Adding a Custom Theme* on page 60
- *Setting the Application Theme* on page 53
- *Creating a New Project* on page 33

Deployment

AppBuilder features several options for packaging your application for specified platforms.

The AppBuilder Deployment menu provides all of the task items that you can perform to:

- create a local Cordova project from your AppBuilder project
- configure and package your application for deployment to SAP Mobile Platform Server as a Kapsel Application
- launch the application in a simulator (iOS or Android) to test application functionality
- publish your application to a Project directory (to make updates to existing Cordova projects)

Create a Local Cordova Project

Generate a Cordova application package for iOS and Android.

Prerequisites

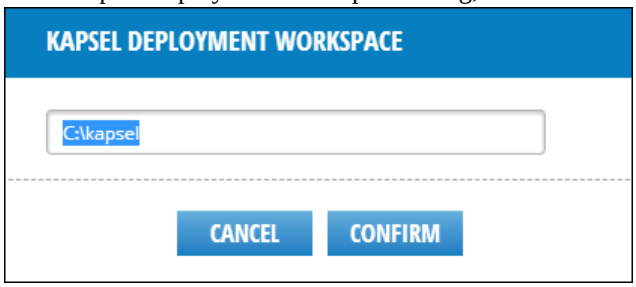
Apache Cordova ships with a set of command-line tools you can use to create, build, and launch an emulator with a single command. AppBuilder supports the command-line tools to create Cordova projects. AppBuilder generates a ZIP file that contains the Web content for Cordova, as well as scripts to create the Cordova project and add SAP Mobile Platform SDK Kapsel plug-ins. Prior to creating your Cordova project, you must:

- Set up your Cordova environment following the steps in *Setting Up the Development Environment > Developing with Cordova*
- If developing on a Mac, set up your Mac environment following the steps in *Setting Up the Development Environment > Setting Up the Mac Environment*.
- Set up your Android development environment following the steps in *Setting Up the Development Environment > Developing for the Android Platform*.
- Create an AppBuilder project as discussed in *Developing Apps with AppBuilder*, *Developing a SuperList App With AppBuilder*, and *Developing a Chart App With AppBuilder*.
- Register your application with SAP Mobile Platform Server using the SAP Management Cockpit administration user interface. See the *SAP Mobile Platform documentation* for more information.

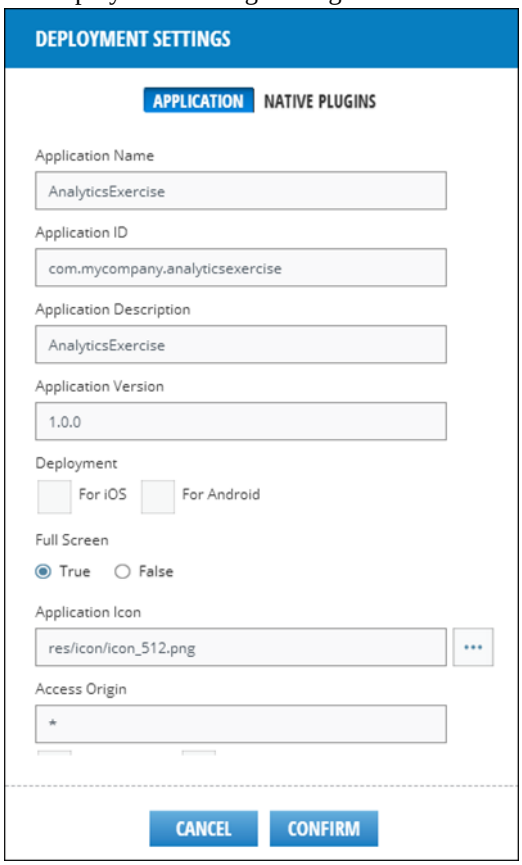
Task

1. Open the application you wish to use to create your Cordova project.

2. Configure the WorkSpace Settings. The WorkSpace is the local folder where your Cordova project will be generated:
   a. In the AppBuilder menu, select **Deployments** > **Workspace Settings**.
   b. In the Kapsel Deployment Workspace dialog, enter a valid local folder name as shown:



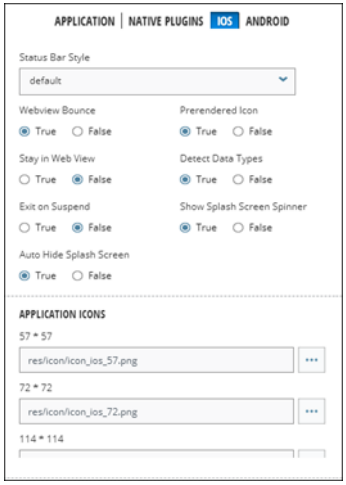
   c. Click **Confirm** to save your settings and close the dialog.
3. In the AppBuilder menu, select **Deployment** > **Create Cordova Project**. This will open the Deployment Settings dialog:



- Application name – the name of the application.
- Application ID – If you are using a SAP Mobile Platform data source or Kapsel plug-ins in the application, this must match the SAP Mobile Platform settings.

  If you are not using any SAP Mobile Platform or Kapsel plug-ins, this can be any application ID that is valid for Cordova.
- (Optional) Application description – a description for your application.
- Application version – this is the same as the version in project settings.
- Deployment – you can choose one or both platforms.
  - For iOS – if you select this, an iOS settings tab is shown, in which you can choose iOS specific settings.
  - For Android – if you select this, an Androd settings tab is shown in which you can choose Android specific settings.
- Application icon – choose the icon for the application. AppBuilder provides some default icons in the `/res/icon` folder in the Project Explorer for the sample applications that are included. You can upload your own icons as .png files by clicking **File > Upload File** in the AppBuilder menu.

4. Click **Native Plugins** to add plugins to your device:
   a. Click the **Cordova** tab to select the Cordova plugins you would like to include in your project:
      1. Enter the path to the Cordova plugins. Leave this field blank if you are using the global Cordova path.
      2. Click the checkbox(es) next to the plugin(s) you with to use in your project.
   b. Click the **Kapsel** tab to select the Kapsel SDK plugins you would like to include in your project
      1. Enter the path to the Kapsel SDK plugins. On Windows, the Kapsel plugins are located in the `\KapselSDK\plugins` folder in the MobileSDK3 install directory. For example, if SAP Mobile Platform SDK is installed in `C:\SAP`, then the KapselSDK plugin path is `C:\SAP\MobileSDK3\KapselSDK\plugins`. On Mac, the Kapsel plugins are located in the directory the `MobileSDK.zip` file is extracted. For example, if the `MobileSDK.zip` is extracted to `%HOME%/MobileSDK3`, the Kapsel plugin path would be `/Users/<logon user>/MobileSDK3/KapselSDK/plugins`.
      2. Click the checkbox(es) next to the plugin(s) you wish to use in your project.
   c. (Optional) Click the **Other Plugins** tab to enter plugin paths for other 3rd party plugins you wish to use in your project.
      1. Click the **+** add the plugin path to be used in your project. You can add additional plugins, remove plugins, or change the order of the plugins list using the navigation buttons to the right of the plugin path field.
5. (Optional) Click the **iOS** tab to configure the iOS device simulator settings. This option will be available if you selected the **For iOS** checkbox in the **Application** tab.



Select your desired settings and icons for the application.

6. (Optional) Click the **Android** tab to configure the Android device simulator settings. This option will be available if you selected the **For Android** checkbox in the **Application** tab.

Select your desired settings and icons for the application.

7. Click **Confirm** to save your settings and create your project. AppBuilder creates the local Cordova project in the Workspace setting folder you specified in Step 2.

See also

- *Deploying a Kapsel App on SAP Mobile Platform Server* on page 69
- *Launch to Simulator* on page 70
- *Setting Up the Development Environment* on page 3

Deploying a Kapsel App on SAP Mobile Platform Server

Deploy a local Cordova project as a Kapsel application to SAP Mobile Server.

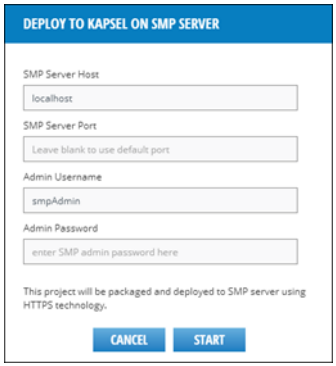
Prerequisites

Kapsel is a set of SAP plugins for Apache Cordova. When you deploy a local Cordova project with Kapsel plugins to SAP Mobile Platform Server, you can leverage capabilities such as application life cycle management, implementation of a common logon manager and single sign-on (SSO), and integration with SAP Mobile Platform Server-based push notifications. AppBuilder provides scripts to consume the Kapsel plugin APIs in your AppBuilder applications. Prior to deploying your Cordova project as a Kapsel application, you must:

- Set up your Cordova environment following the steps in *Setting Up the Development Environment > Developing with Cordova*
- If developing on a Mac, set up your Mac environment following the steps in *Setting Up the Development Environment > Setting Up the Mac Environment*.
- Set up your Android development environment following the steps in *Setting Up the Development Environment > Developing for the Android Platform*.
- Create an AppBuilder project as discussed in *Developing Apps with AppBuilder*, *Developing a SuperList App With AppBuilder*, and *Developing a Chart App With AppBuilder*.
- Register your application with SAP Mobile Platform Server using the SAP Management Cockpit administration user interface. See the *SAP Mobile Platform documentation* for more information.
- Generate a local Cordova project for the native iOS and/or Android platform(s). See *Create a Local Cordova Project*.

Task

1. Open the application you wish to deploy as a Kapsel Application.
2. In the AppBuilder menu, select **Deployment > Kapsel on SMPServer**.
3. In the **Deploy to SMP Server** dialog, enter the information, and click **Start**:



- SMP Server Host - the IP address or path to the SAP Mobile Platform Server host.
- SMP Server Port - the port for the SAP Mobile Platform Server. If the admin port is the default 8083, leave this field blank.
- Admin Username - the administrative user name for SAP Mobile Platform Server.
- Admin Password - the administrative password for SAP Mobile Platform Server.

4. Click Start to deploy the project to SAP Mobile Platform Server. After successful deployment, you can verify your application using the Management Cockpit in SAP Mobile Platform Server. For more information about managing your application in SAP Mobile Platform Server using Management Cockpit, see the *SAP Mobile Platform Server* documentation.

See also

- *Create a Local Cordova Project* on page 65
- *Launch to Simulator* on page 70
- *Setting Up the Development Environment* on page 3

Launch to Simulator

Launch an AppBuilder application in a platform simulator.

Prerequisites

AppBuilder allows you to fully test the functionality of your Cordova application for both the iOS and Android platforms by launching the application on a device simulator. Prior to launching your application, you must:

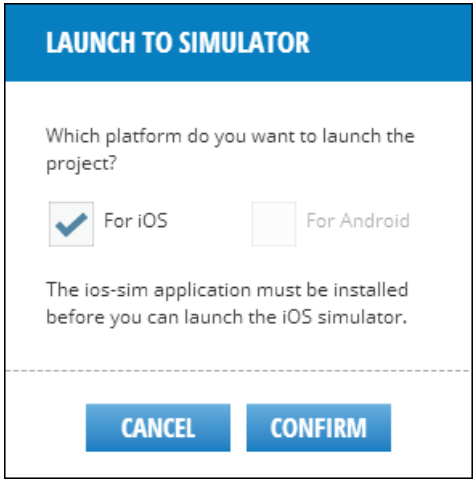
- If developing on a Mac, set up your Mac environment following the steps in *Setting Up the Development Environment* > *Setting Up the Mac Environment*.

  Note: You must have ios-sim installed to allow the Cordova command line to start the iOS simulator on Mac.

- Set up your Android development environment following the steps in *Setting Up the Development Environment* > *Developing for the Android Platform*.
- Create an AppBuilder project as discussed in *Developing Apps with AppBuilder*, *Developing a SuperList App With AppBuilder*, and *Developing a Chart App With AppBuilder*.
- Generate a local Cordova project for the native iOS and/or Android platform(s). See *Create a Local Cordova Project*.
- If testing an application deployed as a Kapsel application to SAP Mobile Platform Server, you must complete the steps in *Create a Local Cordova Project* and *Deploying a Kapsel App on SAP Mobile Platform*.

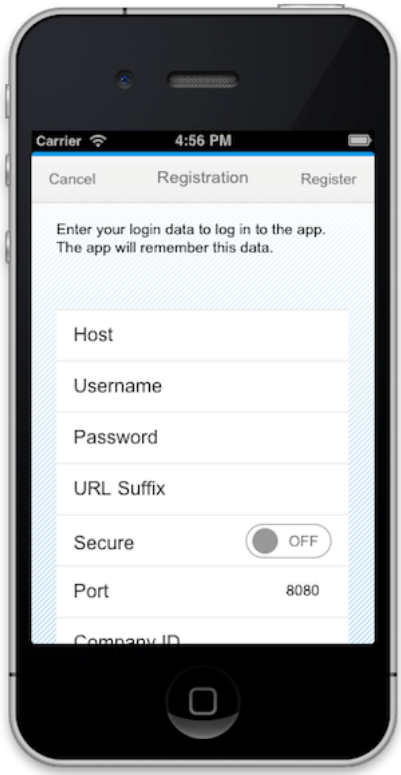
Task

1. Open the application you wish to launch.
2. In the AppBuilder menu, select **Deployment > Launch to Simulator**. The **Launch to Simulator** dialog opens:

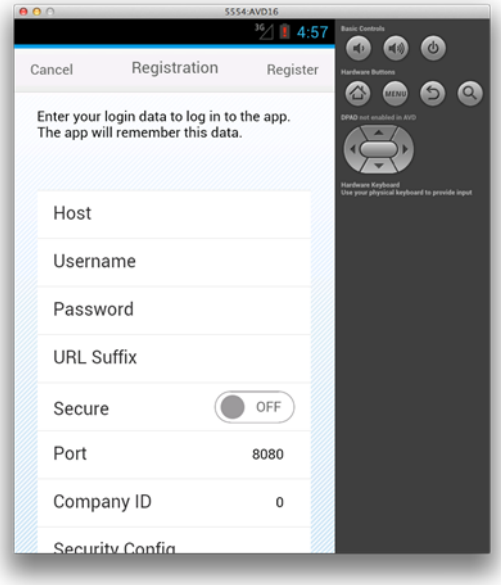


3. In the **Launch to Simulator** dialog, click the checkbox(es) to select for which platform(s) to launch your application, and click **Confirm**. Upon successful launch, your application will appear in the appropriate platform simulator(s):

   Example of application launched in iOS Simulator:



Example of application launched in Android simulator:



4. Test the functionality of your application within the simulator, using your mouse to simulate touch- and key-actions.

See also

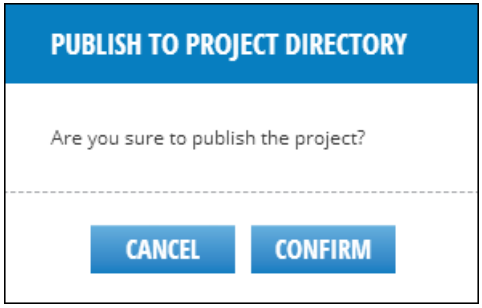
- *Create a Local Cordova Project* on page 65
- *Deploying a Kapsel App on SAP Mobile Platform Server* on page 69
- *Setting Up the Development Environment* on page 3

Publish to Project Directory

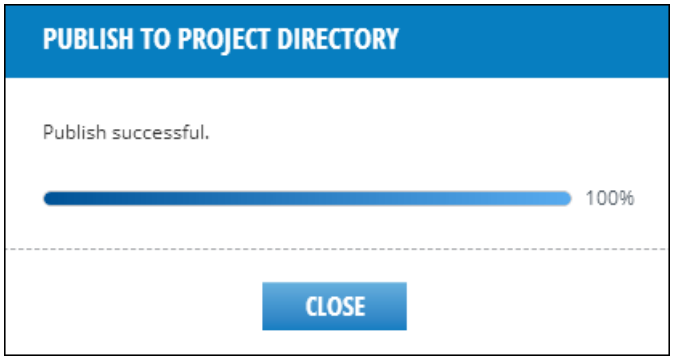
Publish application updates to a Project directory.

Once you have created an application, you can make updates and then republish those updates to an AppBuilder project. The updates will then be reflected in the project directory in `<AppBuilder_Home>\ares-project\hermes\filesystem\root\guest\<Project_Name>`, or in the case of a local Cordova project, the changes will be reflected in the workspace that you designated when you created the project. See *Create a Local Cordova Project*.

1. Open the application you wish to use to update or modify.
2. Modify your application and save your changes.
3. In the AppBuilder menu, select **Deployment** > **Publish to Project Directory**. This will open the Publish to Project Directory dialog:



4. Click **Confirm** to begin the project build. The publish process will overwrite any existing project files. While the publish process is completing, a progress bar is displayed in the Publish to Project Directory dialog. Once successful, a success message and a full progress bar are displayed:



Once you have published updates to your project, you can test functionality using the steps in *Launch to Simulator*. If your project was previously deployed to SAP Mobile Platform Server, you will need to re-deploy the application to push updates to users. Follow the steps listed in *Deploying a Kapsel App on SAP Mobile Platform Server*.

Logging

AppBuilder provides logging features that you can leverage to enable or disable logging and customize log levels, log file size, and backup.

AppBuilder logs are generated in the `<AppBuilder_Home>\ares-project\logs\appbuilder.log` file. Settings for the appbuilder.log file are managed in the `ide.json` script file in the `<AppBuilder_Home>\ares-project` directory. By modifying the `ide.json`, you can customize how AppBuilder logs information. To enable or disable logging in AppBuilder:

1. Navigate to `<AppBuilder_Home>\ares-project\` and open the ide.json file with a text editor.
   - Change `"enabled": "true",` to `"enabled": "false",` to disable logging. When you launch AppBuilder again, no entries will be made in `appbuilder.log` until you enable logging again.
   - Change `"enabled": "false",` to `"enabled": "true",` to enable logging. When you launch AppBuilder again, `appbuilder.log` will begin logging information according to the settings you have set in the `ide.json` file.
2. Save your changes and exit the file.

You can also customize other logging features within AppBuilder by modifying additional settings in the `ide.json` file.

Setting the Log Level

Set the log level for AppBuilder.

1. Navigate to `<AppBuilder_Home>\ares-project\` and open the ide.json file with a text editor.
2. Edit the log level in the file.

```
"log": {
        "enabled": "true",
        "level": "INFO",
        "size": 500000,
        "backup": 5
```

Supported log priority levels include:

```
ALL: new Level(Number.MIN_VALUE, "ALL")
  , INFO: new Level(20000, "INFO")
  , WARN: new Level(30000, "WARN")
  , ERROR: new Level(40000, "ERROR")
```

```
  , OFF: new Level(Number.MAX_VALUE, "OFF")
  };
```

3. Save the file.

   When you review the appbuilder.log file, only information designated within the specified log priority levels will be displayed.

See also

- *Clearing the Logging Information* on page 76
- *Specifying Log Backup Information* on page 76

Clearing the Logging Information

Clear the log file.

Log files are backed up automatically according to the settings of log size and backup number in the `<AppBuilder_Home>\ares-project\ide.json` file:

```
"log": {
        "enabled": "true",
        "level": "INFO",
        "size": 500000,
        "backup": 5
```

1. To clear the log file, go to `<AppBuilder_Home>\ares-project\logs\` and open the `appbuilder.log` with a text editor.
2. Clear the information.
3. Save the file.

See also

- *Setting the Log Level* on page 75
- *Specifying Log Backup Information* on page 76

Specifying Log Backup Information

Specify the size and number of backup log files to maintain.

Log files are backed up automatically according to the settings of log size and backup number in the `<AppBuilder_Home>\ares-project\ide.json` file:

```
"log": {
        "enabled": "true",
        "level": "INFO",
        "size": 500000,
        "backup": 5
```

1. To modify the size and number of backup files, navigate to `<AppBuilder_Home>\ares-project` and open the `ide.json` with a text editor.
2. Enter the size of the file. `"size": <size in bytes>,`. For example, 10 kb would be entered as: `"size": 10000,`
3. Enter the number of backups. `"backup": <number of backups>`. For example, to maintain 10 backups, you would enter: `"backup": 10`.
4. Save the file.

See also

- *Setting the Log Level* on page 75
- *Clearing the Logging Information* on page 76

AppBuilder API Reference

Use the AppBuilder API Reference as the primary reference for all API listings.

sap.apb.makit.Chart class

MAKit Chart

Version: 1.0

Syntax

new `Chart`([*sId*], [*mSettings*])

Constructor for a new Chart. Accepts an object literal <code>mSettings</code> that defines initial property values, aggregated and associated objects as well as event handlers. If the name of a setting is ambiguous (e.g. a property has the same name as an event), then the framework assumes property, aggregation, association, event in that order. To override this automatic resolution, one of the prefixes "aggregation:", "association:" or "event:" can be added to the name of the setting (such a prefixed name must be enclosed in single or double quotes). The supported settings are:

- Properties
  - #getType type : sap.apb.makit.ChartType (default: sap.apb.makit.ChartType.Column)
  - #getCategorySortOrder categorySortOrder : sap.apb.makit.SortOrder (default: sap.apb.makit.SortOrder.None)
  - #getDataSource dataSource : string
  - height : sap.ui.core.CSSSize (default: '100%')
  - #getLegendPosition legendPosition : sap.apb.makit.LegendPosition (default: sap.apb.makit.LegendPosition.None)
  - #getLineThickness lineThickness : float (default: 1)
  - #getMaxSliceCount maxSliceCount : int (default: 12)
  - #getMetadataFile metadataFile : sap.ui.core.URI
  - #getShowRangeSelector showRangeSelector : boolean (default: true)
  - #getShowTableView showTableView : boolean (default: false)
  - #getShowTableValue showTableValue : boolean (default: true)
  - #getShowToolbar showToolbar : boolean (default: true)
  - #getWidth width : sap.ui.core.CSSSize (default: '100%')
- Aggregations
- Associations
- Events

- sap.apb.makit.Chart#event:doubletap doubletap : fnListenerFunction or [fnListenerFunction, oListenerObject] or [oData, fnListenerFunction, oListenerObject]
- sap.apb.makit.Chart#event:tap tap : fnListenerFunction or [fnListenerFunction, oListenerObject] or [oData, fnListenerFunction, oListenerObject]

Extends

- sap.ui.core.Control

Parameters

Name	Type	Argument	Description
sId	string	(optional)	id for the new control, generated automatically if no id is given
mSettings	object	(optional)	initial settings for the new control

Methods

Name	Description
attachDoubletap([oData], fnFunction, [oListener]) on page 83	Attach event handler <code>fnFunction</code> to the 'doubletap' event of this <code>sap.apb.makit.Chart</code>.
attachTap([oData], fnFunction, [oListener]) on page 84	Attach event handler <code>fnFunction</code> to the 'tap' event of this <code>sap.apb.makit.Chart</code>.
detachDoubletap(fnFunction, oListener) on page 85	Detach event handler <code>fnFunction</code> from the 'doubletap' event of this <code>sap.apb.makit.Chart</code>.
detachTap(fnFunction, oListener) on page 86	Detach event handler <code>fnFunction</code> from the 'tap' event of this <code>sap.apb.makit.Chart</code>.
extend(sClassName, [oClassInfo], [FNMetaImpl]) on page 87	
fireDoubletap([mArguments]) on page 87	Fire event doubletap to attached listeners.
fireTap([mArguments]) on page 88	Fire event tap to attached listeners.

getCategorySortOrder() on page 88	Getter for property <code>categorySortOrder</code>. Sort order for category.
getDataSource() on page 89	Getter for property <code>dataSource</code>. dataSource name of the chart. Default value is empty/<code>undefined</code>
getHeight() on page 89	Getter for property <code>height</code>. The height of the Chart. Default value is <code>100%</code>
getLegendPosition() on page 90	Getter for property <code>legendPosition</code>. Legend position all chart types except Bar chart.
getLineThickness() on page 90	Getter for property <code>lineThickness</code>. Specify the line thickness of the line graph.
getMaxSliceCount() on page 91	Getter for property <code>maxSliceCount</code>. Set the maximum number of slices in a Pie/Donut chart.
getMetadataFile() on page 91	Getter for property <code>metadataFile</code>. Metadata file URI that is assigned to this chart.
getNumberOfCategories() on page 91	Get the number of distinct category values.
getSelectedCategory() on page 92	Get the value of the currently highlighted category.
getSelectedSeries() on page 92	Get the value of the currently highlighted series.
getShowRangeSelector() on page 93	Getter for property <code>showRangeSelector</code>. Specify whether the range selector should be visible. Default value is <code>true</code>
getShowTableValue() on page 93	Getter for property <code>showTableValue</code>. Toggle to display the table value on a Bar chart.
getShowTableView() on page 93	Getter for property <code>showTableView</code>. Toggle to display table view. Default value is <code>false</code>
getShowToolbar() on page 94	Getter for property <code>showToolbar</code>. Show or hide the chart's toolbar. Default value is <code>true</code>

getType() on page 94	Getter for property <code>type</code>. Chart type. Default value is <code>Column</code>
getWidth() on page 95	Getter for property <code>width</code>. The width of the Chart. Default value is <code>100%</code>
refreshData() on page 95	Re-retrieve data from the datasource and re-render chart.
setCategorySortOrder(oCategorySortOrder) on page 95	Setter for property <code>categorySortOrder</code>. Default value is <code>None</code>
setDataSource(sDataSource) on page 96	Setter for property <code>dataSource</code>. Default value is empty/<code>undefined</code>
setHeight(sHeight) on page 96	Setter for property <code>height</code>. Default value is <code>100%</code>
setLegendPosition(oLegendPosition) on page 97	Setter for property <code>legendPosition</code>. Default value is <code>None</code>
setLineThickness(fLineThickness) on page 97	Setter for property <code>lineThickness</code>. Default value is <code>1</code>
setMaxSliceCount(iMaxSliceCount) on page 98	Setter for property <code>maxSliceCount</code>. Default value is <code>12</code>
setMetadataFile(sMetadataFile) on page 98	Setter for property <code>metadataFile</code>. Default value is empty/<code>undefined</code>
setShowRangeSelector(bShowRangeSelector) on page 99	Setter for property <code>showRangeSelector</code>. Default value is <code>true</code>
setShowTableValue(bShowTableValue) on page 100	Setter for property <code>showTableValue</code>. Default value is <code>true</code>
setShowTableView(bShowTableView) on page 100	Setter for property <code>showTableView</code>. Default value is <code>false</code>
setShowToolbar(bShowToolbar) on page 101	Setter for property <code>showToolbar</code>. Default value is <code>true</code>
setType(oType) on page 101	Setter for property <code>type</code>. Default value is <code>Column</code>
setWidth(sWidth) on page 102	Setter for property <code>width</code>. Default value is <code>100%</code>

Events

Name	Description
doubletap(oControlEvent, oControlEvent.get-Source, oControlEvent.getParameters) on page 102	Double tap event on chart.
tap(oControlEvent, oControlEvent.getSource, oControlEvent.getParameters) on page 103	Single tap event on the chart.

Source
makit/Chart.API.js, line 11 on page 204.

attachDoubletap([oData], fnFunction, [oListener]) method

Attach event handler <code>fnFunction</code> to the 'doubletap' event of this <code>sap.apb.makit.Chart</code>.

. When called, the context of the event handler (its <code>this</code>) will be bound to <code>oListener<code> if specified otherwise to this <code>sap.apb.makit.Chart</code>.

itself. Double tap event on chart.

Syntax
`attachDoubletap`([*oData*], *fnFunction*, [*oListener*]) {sap.apb.makit.Chart}

Parameters

Name	Type	Argument	Default	Description
oData	object	(optional)		An application specific payload object, that will be passed to the event handler along with the event object when firing the event.
fnFunction	function			The function to call, when the event occurs.

oListener	object	(optional)	this	Context object to call the event handler with.Defaults to this <code>sap.apb.makit.Chart</code>. itself.

Returns
<code>this</code> to allow method chaining

Type:

sap.apb.makit.Chart on page 79

Source
makit/Chart.API.js, line 187 on page 211.

attachTap([oData], fnFunction, [oListener]) method

Attach event handler <code>fnFunction</code> to the 'tap' event of this <code>sap.apb.makit.Chart</code>.

. When called, the context of the event handler (its <code>this</code>) will be bound to <code>oListener<code> if specified otherwise to this <code>sap.apb.makit.Chart</code>.

itself. Single tap event on the chart.

Syntax
`attachTap`([*oData*], *fnFunction*, [*oListener*]) {sap.apb.makit.Chart}

Parameters

Name	Type	Argument	Default	Description

oData	object	(optional)		An application specific payload object, that will be passed to the event handler along with the event object when firing the event.
fnFunction	function			The function to call, when the event occurs.
oListener	object	(optional)	this	Context object to call the event handler with.Defaults to this <code>sap.apb.makit.Chart</code>. itself.

Returns
<code>this</code> to allow method chaining

Type:

sap.apb.makit.Chart on page 79

Source
makit/Chart.API.js, line 196 on page 211.

detachDoubletap(fnFunction, oListener) method

Detach event handler <code>fnFunction</code> from the 'doubletap' event of this <code>sap.apb.makit.Chart</code>.

The passed function and listener object must match the ones used for event registration.

Syntax
`detachDoubletap`(*fnFunction*, *oListener*) {sap.apb.makit.Chart}

Parameters

Name	Type	Description
fnFunction	function	The function to call, when the event occurs.
oListener	object	Context object on which the given function had to be called.

Returns

<code>this</code> to allow method chaining

Type:

sap.apb.makit.Chart on page 79

Source

makit/Chart.API.js, line 189 on page 211.

detachTap(fnFunction, oListener) method

Detach event handler <code>fnFunction</code> from the 'tap' event of this <code>sap.apb.makit.Chart</code>.

The passed function and listener object must match the ones used for event registration.

Syntax

`detachTap(` *fnFunction*, *oListener* `)` {sap.apb.makit.Chart}

Parameters

Name	Type	Description
fnFunction	function	The function to call, when the event occurs.
oListener	object	Context object on which the given function had to be called.

Returns

<code>this</code> to allow method chaining

Type:

sap.apb.makit.Chart on page 79

Source
makit/Chart.API.js, line 198 on page 211.

extend(sClassName, [oClassInfo], [FNMetaImpl]) method

Creates a new subclass of class sap.apb.makit.Chart with name <code>sClassName</code> and enriches it with the information contained in <code>oClassInfo</code>. <code>oClassInfo</code> might contain the same kind of informations as described in sap.ui.core.Element.extend Element.extend.

Syntax
<static> `extend`(*sClassName*, [*oClassInfo*], [*FNMetaImpl*]) {function}

Parameters

Name	Type	Argument	Description
sClassName	string		name of the class to be created
oClassInfo	object	(optional)	object literal with informations about the class
FNMetaImpl	function	(optional)	constructor function for the metadata object.If not given, it defaults to sap.ui.core.ElementMetadata.

Returns
the created class / constructor function

Type:

function

Source
makit/Chart.API.js, line 101 on page 208.

fireDoubletap([mArguments]) method

Fire event doubletap to attached listeners.

Syntax
`fireDoubletap`([*mArguments*]) {sap.apb.makit.Chart}

Parameters

Name	Type	Argument	Description
mArguments	Map	(optional)	the arguments to pass along with the event.

Returns

<code>this</code> to allow method chaining

Type:

sap.apb.makit.Chart on page 79

Source

makit/Chart.API.js, line 191 on page 211.

fireTap([mArguments]) method

Fire event tap to attached listeners.

Syntax

`fireTap`([*mArguments*]) ~~{~~sap.apb.makit.Chart}

Parameters

Name	Type	Argument	Description
mArguments	Map	(optional)	the arguments to pass along with the event.

Returns

<code>this</code> to allow method chaining

Type:

sap.apb.makit.Chart on page 79

Source

makit/Chart.API.js, line 200 on page 211.

getCategorySortOrder() method

Getter for property <code>categorySortOrder</code>. Sort order for category.

If None, the category's column is expected to be pre-sorted. Default value is <code>None</code>

Syntax
`getCategorySortOrder()` {sap.apb.makit.SortOrder}

Returns
the value of property <code>categorySortOrder</code>

Type:

sap.apb.makit.SortOrder on page 110

Source
makit/Chart.API.js, line 125 on page 208.

getDataSource() method

Getter for property <code>dataSource</code>. dataSource name of the chart. Default value is empty/<code>undefined</code>

Syntax
`getDataSource()` {string}

Returns
the value of property <code>dataSource</code>

Type:

string

Source
makit/Chart.API.js, line 130 on page 209.

getHeight() method

Getter for property <code>height</code>. The height of the Chart. Default value is <code>100%</code>

Syntax
`getHeight()` {sap.ui.core.CSSSize}

Returns
the value of property <code>height</code>

Type:

sap.ui.core.CSSSize

Source
makit/Chart.API.js, line 135 on page 209.

getLegendPosition() method

Getter for property <code>legendPosition</code>. Legend position all chart types except Bar chart.

Bar chart only support Bottom position. Default value is <code>None</code>

Syntax
`getLegendPosition()` {sap.apb.makit.LegendPosition}

Returns
the value of property <code>legendPosition</code>

Type:

sap.apb.makit.LegendPosition on page 109

Source
makit/Chart.API.js, line 140 on page 209.

getLineThickness() method

Getter for property <code>lineThickness</code>. Specify the line thickness of the line graph.

Only applies to Line chart type. Default value is <code>1</code>

Syntax
`getLineThickness()` {float}

Returns
the value of property <code>lineThickness</code>

Type:

float

Source
makit/Chart.API.js, line 145 on page 209.

getMaxSliceCount() method

Getter for property <code>maxSliceCount</code>. Set the maximum number of slices in a Pie/Donut chart.

If exceeding the specified value, the rest will be categorised into a single slice. Only applies to Pie/Donut. Default value is <code>12</code>

Syntax

`getMaxSliceCount()` {int}

Returns

the value of property <code>maxSliceCount</code>

Type:

int

Source

makit/Chart.API.js, line 150 on page 209.

getMetadataFile() method

Getter for property <code>metadataFile</code>. Metadata file URI that is assigned to this chart.

Metadata is mandatory for a chart to be rendered because it contains the chart's details. Default value is empty/<code>undefined</code>

Syntax

`getMetadataFile()` {sap.ui.core.URI}

Returns

the value of property <code>metadataFile</code>

Type:

sap.ui.core.URI

Source

makit/Chart.API.js, line 155 on page 209.

getNumberOfCategories() method

Get the number of distinct category values.

Syntax
`getNumberOfCategories()` {int}

Returns

Type:

int

Source
makit/Chart.API.js, line 209 on page 211.

getSelectedCategory() method

Get the value of the currently highlighted category.

Syntax
`getSelectedCategory()` {string}

Returns

Type:

string

Source
makit/Chart.API.js, line 203 on page 211.

getSelectedSeries() method

Get the value of the currently highlighted series.

Syntax
`getSelectedSeries()` {string}

Returns

Type:

string

Source
makit/Chart.API.js, line 206 on page 211.

getShowRangeSelector() method

Getter for property <code>showRangeSelector</code>. Specify whether the range selector should be visible. Default value is <code>true</code>

Syntax

`getShowRangeSelector()` {boolean}

Returns

the value of property <code>showRangeSelector</code>

Type:

boolean

Source

makit/Chart.API.js, line 160 on page 210.

getShowTableValue() method

Getter for property <code>showTableValue</code>. Toggle to display the table value on a Bar chart.

Only applies to Bar chart. Default value is <code>true</code>

Syntax

`getShowTableValue()` {boolean}

Returns

the value of property <code>showTableValue</code>

Type:

boolean

Source

makit/Chart.API.js, line 170 on page 210.

getShowTableView() method

Getter for property <code>showTableView</code>. Toggle to display table view. Default value is <code>false</code>

Syntax

`getShowTableView()` {boolean}

Returns
the value of property <code>showTableView</code>

Type:

boolean

Source
makit/Chart.API.js, line 165 on page 210.

getShowToolbar() method

Getter for property <code>showToolbar</code>. Show or hide the chart's toolbar. Default value is <code>true</code>

Syntax
`getShowToolbar()` {boolean}

Returns
the value of property <code>showToolbar</code>

Type:

boolean

Source
makit/Chart.API.js, line 175 on page 210.

getType() method

Getter for property <code>type</code>. Chart type. Default value is <code>Column</code>

Syntax
`getType()` {sap.apb.makit.ChartType}

Returns
the value of property <code>type</code>

Type:

sap.apb.makit.ChartType on page 103

Source
makit/Chart.API.js, line 120 on page 208.

getWidth() method

Getter for property <code>width</code>. The width of the Chart. Default value is <code>100%</code>

Syntax

`getWidth()` {sap.ui.core.CSSSize}

Returns

the value of property <code>width</code>

Type:

sap.ui.core.CSSSize

Source

makit/Chart.API.js, line 180 on page 210.

refreshData() method

Re-retrieve data from the datasource and re-render chart.

Syntax

`refreshData()` {void}

Returns

Type:

void

Source

makit/Chart.API.js, line 212 on page 211.

setCategorySortOrder(oCategorySortOrder) method

Setter for property <code>categorySortOrder</code>. Default value is <code>None</code>

Syntax

`setCategorySortOrder(` *oCategorySortOrder* `)` {sap.apb.makit.Chart}

Parameters

Name	Type	Description

oCategorySortOrder	*sap.apb.makit.SortOrder* on page 110	new value for property <code>categorySortOrder</code>

Returns
<code>this</code> to allow method chaining

Type:

sap.apb.makit.Chart on page 79

Source
makit/Chart.API.js, line 127 on page 209.

setDataSource(sDataSource) method

Setter for property <code>dataSource</code>. Default value is empty/<code>undefined</code>

Syntax
`setDataSource`(*sDataSource*) {sap.apb.makit.Chart}

Parameters

Name	Type	Description
sDataSource	string	new value for property <code>dataSource</code>

Returns
<code>this</code> to allow method chaining

Type:

sap.apb.makit.Chart on page 79

Source
makit/Chart.API.js, line 132 on page 209.

setHeight(sHeight) method

Setter for property <code>height</code>. Default value is <code>100%</code>

Syntax
`setHeight`(*sHeight*) {sap.apb.makit.Chart}

Parameters

Name	Type	Description
sHeight	sap.ui.core.CSSSize	new value for property <code>height</code>

Returns
<code>this</code> to allow method chaining

Type:

sap.apb.makit.Chart on page 79

Source
makit/Chart.API.js, line 137 on page 209.

setLegendPosition(oLegendPosition) method

Setter for property <code>legendPosition</code>. Default value is <code>None</code>

Syntax
`setLegendPosition`(*oLegendPosition*) {sap.apb.makit.Chart}

Parameters

Name	Type	Description
oLegendPosition	*sap.apb.makit.LegendPosition* on page 109	new value for property <code>legendPosition</code>

Returns
<code>this</code> to allow method chaining

Type:

sap.apb.makit.Chart on page 79

Source
makit/Chart.API.js, line 142 on page 209.

setLineThickness(fLineThickness) method

Setter for property <code>lineThickness</code>. Default value is <code>1</code>

Syntax
`setLineThickness`(*fLineThickness*) {sap.apb.makit.Chart}

Parameters

Name	Type	Description
fLineThickness	float	new value for property <code>lineThickness</code>

Returns
<code>this</code> to allow method chaining

Type:

sap.apb.makit.Chart on page 79

Source
makit/Chart.API.js, line 147 on page 209.

setMaxSliceCount(iMaxSliceCount) method

Setter for property <code>maxSliceCount</code>. Default value is <code>12</code>

Syntax
`setMaxSliceCount(` *iMaxSliceCount* `)` {sap.apb.makit.Chart}

Parameters

Name	Type	Description
iMaxSliceCount	int	new value for property <code>maxSliceCount</code>

Returns
<code>this</code> to allow method chaining

Type:

sap.apb.makit.Chart on page 79

Source
makit/Chart.API.js, line 152 on page 209.

setMetadataFile(sMetadataFile) method

Setter for property <code>metadataFile</code>. Default value is empty/<code>undefined</code>

Syntax

`setMetadataFile(` *sMetadataFile* `)` {sap.apb.makit.Chart}

Parameters

Name	Type	Description
sMetadataFile	sap.ui.core.URI	new value for property <code>metadataFile</code>

Returns

<code>this</code> to allow method chaining

Type:

sap.apb.makit.Chart on page 79

Source

makit/Chart.API.js, line 157 on page 210.

setShowRangeSelector(bShowRangeSelector) method

Setter for property <code>showRangeSelector</code>. Default value is <code>true</code>

Syntax

`setShowRangeSelector(` *bShowRangeSelector* `)` {sap.apb.makit.Chart}

Parameters

Name	Type	Description
bShowRangeSelector	boolean	new value for property <code>showRangeSelector</code>

Returns

<code>this</code> to allow method chaining

Type:

sap.apb.makit.Chart on page 79

Source

makit/Chart.API.js, line 162 on page 210.

setShowTableValue(bShowTableValue) method

Setter for property <code>showTableValue</code>. Default value is <code>true</code>

Syntax

`setShowTableValue(` *bShowTableValue* `)` {sap.apb.makit.Chart}

Parameters

Name	Type	Description
bShowTableValue	boolean	new value for property <code>showTableValue</code>

Returns

<code>this</code> to allow method chaining

Type:

sap.apb.makit.Chart on page 79

Source

makit/Chart.API.js, line 172 on page 210.

setShowTableView(bShowTableView) method

Setter for property <code>showTableView</code>. Default value is <code>false</code>

Syntax

`setShowTableView(` *bShowTableView* `)` {sap.apb.makit.Chart}

Parameters

Name	Type	Description
bShowTableView	boolean	new value for property <code>showTableView</code>

Returns

<code>this</code> to allow method chaining

Type:

sap.apb.makit.Chart on page 79

Source
makit/Chart.API.js, line 167 on page 210.

setShowToolbar(bShowToolbar) method

Setter for property <code>showToolbar</code>. Default value is <code>true</code>

Syntax
`setShowToolbar(` *bShowToolbar* `)` {sap.apb.makit.Chart}

Parameters

Name	Type	Description
bShowToolbar	boolean	new value for property <code>showToolbar</code>

Returns
<code>this</code> to allow method chaining

Type:

sap.apb.makit.Chart on page 79

Source
makit/Chart.API.js, line 177 on page 210.

setType(oType) method

Setter for property <code>type</code>. Default value is <code>Column</code>

Syntax
`setType(` *oType* `)` {sap.apb.makit.Chart}

Parameters

Name	Type	Description
oType	*sap.apb.makit.ChartType* on page 103	new value for property <code>type</code>

Returns
<code>this</code> to allow method chaining

Type:

sap.apb.makit.Chart on page 79

Source
makit/Chart.API.js, line 122 on page 208.

setWidth(sWidth) method

Setter for property <code>width</code>. Default value is <code>100%</code>

Syntax
`setWidth(` *sWidth* `)` {sap.apb.makit.Chart}

Parameters

Name	Type	Description
sWidth	sap.ui.core.CSSSize	new value for property <code>width</code>

Returns
<code>this</code> to allow method chaining

Type:

sap.apb.makit.Chart on page 79

Source
makit/Chart.API.js, line 182 on page 210.

doubletap(oControlEvent, oControlEvent.getSource, oControlEvent.getParameters) event

Double tap event on chart.

Parameters

Name	Type	Description
oControlEvent	sap.ui.base.Event	
oControlEvent.getSource	sap.ui.base.EventProvider	
oControlEvent.getParameters	object	

Source
makit/Chart.API.js, line 185 on page 210.

tap(oControlEvent, oControlEvent.getSource, oControlEvent.getParameters) event

Single tap event on the chart.

Parameters

Name	Type	Description
oControlEvent	sap.ui.base.Event	
oControlEvent.getSource	sap.ui.base.EventProvider	
oControlEvent.getParameters	object	

Source
makit/Chart.API.js, line 194 on page 211.

sap.apb.makit.ChartType class

Enumeration for chart type

Version: 1.0

Syntax
new `ChartType`()

Since
1.0

Source
makit/ChartType.ENUM.js, line 17 on page 226.

Bar member

Horizontal table bar chart

Syntax
<static> `Bar`

Source
makit/ChartType.ENUM.js, line 41 on page 226.

Bubble member

Bubble chart

Syntax
<static> `Bubble`

Source
makit/ChartType.ENUM.js, line 35 on page 226.

Column member

Column chart

Syntax
<static> `Column`

Source
makit/ChartType.ENUM.js, line 23 on page 226.

Line member

Line chart

Syntax
<static> `Line`

Source
makit/ChartType.ENUM.js, line 29 on page 226.

Pie member

Pie chart

Syntax
<static> `Pie`

Source
makit/ChartType.ENUM.js, line 47 on page 227.

sap.DataSourceAPI class

SAP DataSources APIs

Version: 1.0

Syntax

new `DataSourceAPI`()

Methods

Name	Description
decryptPassword([password]) on page 105	Decrypt the specified password and return the decrypted password
encryptPassword([password]) on page 106	Encrypt the specified password and return the encrypted password
setDataSourceInfo([dataSourceName], [user], [password], [url]) on page 106	Change the specified datasource's user name, password, and root URL information
setLogonApplicationContext([smpLogonApplicationContext]) on page 107	Set the application logon context for accesing oData servicesfrom SMP server.
setSMPServerProfile([SMPServer], [SMPApplicationID], [SMPApplicationTag], [allowAnonymous], [user], [password], [connectionID]) on page 108	Change the current SMP Server Profile

Source

common.js, line 31 on page 200.

decryptPassword([password]) method

Decrypt the specified password and return the decrypted password

Syntax

`decryptPassword`([*password*])

Parameters

Name	Type	Argument	Description

password	string	(optional)	The password that is going to be decrypted

Source
common.js, line 99 on page 203.

encryptPassword([password]) method

Encrypt the specified password and return the encrypted password

Syntax
`encryptPassword`([*password*])

Parameters

Name	Type	Argument	Description
password	string	(optional)	The password that is going to be encrypted

Source
common.js, line 87 on page 203.

setDataSourceInfo([dataSourceName], [user], [password], [url]) method

Change the specified datasource’s user name, password, and root URL information

Syntax
`setDataSourceInfo`([*dataSourceName*], [*user*], [*password*], [*url*])

Parameters

Name	Type	Argument	Description
dataSourceName	string	(optional)	The datasource's name for example "OData-Service.Employ-eeDB", it can’t be the table source name

user	string	(optional)	The new user name ,if you don't want to change the user name, you can set it to undefined
password	string	(optional)	The new password, if you don't want to change the password, you can set it to undefined
url	string	(optional)	The new root URL for the data source, if you don't want to change the root URL, you can set it to undefined

Source
common.js, line 54 on page 201.

setLogonApplicationContext([smpLogonApplicationContext]) method

Set the application logon context for accesing oData servicesfrom SMP server.

This context is from SMP Kapsel logon manager.

Syntax
`setLogonApplicationContext( [`*smpLogonApplicationContext*`] )`

Parameters

Name	Type	Argument	Description
smpLogonApplicationContext	string	(optional)	Application login context.This context should be the returned result from the Kapsel logon manager.

Source
common.js, line 42 on page 201.

setSMPServerProfile([SMPServer], [SMPApplicationID], [SMPApplicationTag], [allowAnonymous], [user], [password], [connectionID]) method

Change the current SMP Server Profile

Syntax

`setSMPServerProfile`([*SMPServer*], [*SMPApplicationID*], [*SMPApplicationTag*], [*allowAnonymous*], [*user*], [*password*], [*connectionID*])

Parameters

Name	Type	Argument	Description
SMPServer	string	(optional)	The new SMP Server to use.If you don't want to change it, you can set it to undefined
SMPApplicationID	string	(optional)	The new SMP ApplicationID.If you don't want to change it, you can set it to undefined
SMPApplicationTag	string	(optional)	The new SMP ApplicationTag.If you don't want to change it, you can set it to undefined
allowAnonymous	bool	(optional)	You can set it to true if the server allows anonymous access, otherwise, set it false.If you don't want to change it, you can set it to undefined
user	string	(optional)	The new user name.If you don't want to change it, you can set it to undefined

password	string	(optional)	The new password.If you don't want to change it, you can set it to undefined
connectionID	string	(optional)	The the conection ID to be used.If you want to clear the old connection ID, you can set it to empty string ("").

Source
common.js, line 69 on page 202.

sap.apb.makit.LegendPosition class

Enumeration for legend position.

Version: 1.0

Syntax
new `LegendPosition`()

Source
makit/LegendPosition.ENUM.js, line 16 on page 227.

Bottom member

Legend location is on the bottom of the chart

Syntax
<static> `Bottom`

Source
makit/LegendPosition.ENUM.js, line 34 on page 228.

Left member

Legend location is on the left of the chart

Syntax
<static> `Left`

Source
makit/LegendPosition.ENUM.js, line 28 on page 228.

None member

Hide the legend

Syntax
<static> `None`

Source
makit/LegendPosition.ENUM.js, line 46 on page 228.

Right member

Legend location is on the right of the chart

Syntax
<static> `Right`

Source
makit/LegendPosition.ENUM.js, line 40 on page 228.

Top member

Legend location is on the top of the chart

Syntax
<static> `Top`

Source
makit/LegendPosition.ENUM.js, line 22 on page 228.

sap.apb.makit.SortOrder class

Enumeration for sort order.

Version: 1.0

Syntax
new `SortOrder`()

Source
makit/SortOrder.ENUM.js, line 16 on page 229.

Ascending member

Ascending sort

Syntax
<static> `Ascending`

Source
makit/SortOrder.ENUM.js, line 22 on page 229.

Descending member

Descending sort

Syntax
<static> `Descending`

Source
makit/SortOrder.ENUM.js, line 28 on page 230.

None member

No sorting

Syntax
<static> `None`

Source
makit/SortOrder.ENUM.js, line 34 on page 230.

sap.apb.SuperList class

SuperList control

Version: 1.0

Syntax
new `SuperList`([*sId*], [*mSettings*])

Constructor for a new SuperList. Accepts an object literal <code>mSettings</code> that defines initial property values, aggregated and associated objects as well as event handlers. If the name of a setting is ambiguous (e.g. a property has the same name as an event), then the framework assumes property, aggregation, association, event in that order. To override this automatic resolution, one of the prefixes "aggregation:", "association:" or "event:" can be added to the name of the setting (such a prefixed name must be enclosed in single or double quotes). The supported settings are:

- Properties
  - #getDataSource dataSource : string
  - height : sap.ui.core.CSSSize (default: '100%')
  - #getMetadataFile metadataFile : sap.ui.core.URI
  - #getWidth width : sap.ui.core.CSSSize (default: '100%')
  - #getReadRows readRows : string (default: '200')
- Aggregations
- Associations
- Events
  - sap.apb.SuperList#event:dataTableQuery dataTableQuery : fnListenerFunction or [fnListenerFunction, oListenerObject] or [oData, fnListenerFunction, oListenerObject]
  - sap.apb.SuperList#event:rowFocusChanged rowFocusChanged : fnListenerFunction or [fnListenerFunction, oListenerObject] or [oData, fnListenerFunction, oListenerObject]
  - sap.apb.SuperList#event:itemChanged itemChanged : fnListenerFunction or [fnListenerFunction, oListenerObject] or [oData, fnListenerFunction, oListenerObject]
  - sap.apb.SuperList#event:buttonClicked buttonClicked : fnListenerFunction or [fnListenerFunction, oListenerObject] or [oData, fnListenerFunction, oListenerObject]
  - sap.apb.SuperList#event:updateEnd updateEnd : fnListenerFunction or [fnListenerFunction, oListenerObject] or [oData, fnListenerFunction, oListenerObject]
  - sap.apb.SuperList#event:error error : fnListenerFunction or [fnListenerFunction, oListenerObject] or [oData, fnListenerFunction, oListenerObject]

Extends

- sap.ui.core.Control

Parameters

Name	Type	Argument	Description

sId	string	(optional)	id for the new control, generated automatically if no id is given
mSettings	object	(optional)	initial settings for the new control

Methods

Name	Description
attachButtonClicked([oData], fnFunction, [oListener]) on page 116	Attach event handler <code>fnFunction</code> to the 'buttonClicked' event of this <code>sap.apb.SuperList</code>.
attachDataTableQuery([oData], fnFunction, [oListener]) on page 117	Attach event handler <code>fnFunction</code> to the 'dataTableQuery' event of this <code>sap.apb.SuperList</code>.
attachError([oData], fnFunction, [oListener]) on page 119	Attach event handler <code>fnFunction</code> to the 'error' event of this <code>sap.apb.SuperList</code>.
attachItemChanged([oData], fnFunction, [oListener]) on page 120	Attach event handler <code>fnFunction</code> to the 'itemChanged' event of this <code>sap.apb.SuperList</code>.
attachRowFocusChanged([oData], fnFunction, [oListener]) on page 121	Attach event handler <code>fnFunction</code> to the 'rowFocusChanged' event of this <code>sap.apb.SuperList</code>.
attachUpdateEnd([oData], fnFunction, [oListener]) on page 122	Attach event handler <code>fnFunction</code> to the 'updateEnd' event of this <code>sap.apb.SuperList</code>.
deleteRow(iRow) on page 123	Deletes a row.
detachButtonClicked(fnFunction, oListener) on page 124	Detach event handler <code>fnFunction</code> from the 'buttonClicked' event of this <code>sap.apb.SuperList</code>.
detachDataTableQuery(fnFunction, oListener) on page 124	Detach event handler <code>fnFunction</code> from the 'dataTableQuery' event of this <code>sap.apb.SuperList</code>.
detachError(fnFunction, oListener) on page 125	Detach event handler <code>fnFunction</code> from the 'error' event of this <code>sap.apb.SuperList</code>.

detachItemChanged(fnFunction, oListener) on page 126	Detach event handler <code>fnFunction</code> from the 'itemChanged' event of this <code>sap.apb.SuperList</code>.
detachRowFocusChanged(fnFunction, oListener) on page 126	Detach event handler <code>fnFunction</code> from the 'rowFocusChanged' event of this <code>sap.apb.SuperList</code>.
detachUpdateEnd(fnFunction, oListener) on page 127	Detach event handler <code>fnFunction</code> from the 'updateEnd' event of this <code>sap.apb.SuperList</code>.
drillBack() on page 128	Drills back to prior level
drillDown(iRow) on page 128	Drills down to next level.
extend(sClassName, [oClassInfo], [FNMetaImpl]) on page 128	
filter(sExpr) on page 129	Filters the rows.
fireButtonClicked([mArguments]) on page 130	Fire event buttonClicked to attached listeners.
fireDataTableQuery([mArguments]) on page 130	Fire event dataTableQuery to attached listeners.
fireError([mArguments]) on page 131	Fire event error to attached listeners.
fireItemChanged([mArguments]) on page 131	Fire event itemChanged to attached listeners.
fireRowFocusChanged([mArguments]) on page 132	Fire event rowFocusChanged to attached listeners.
fireUpdateEnd([mArguments]) on page 132	Fire event updateEnd to attached listeners.
getCurrentLevel() on page 133	Gets current level.
getDataSource() on page 133	Getter for property <code>dataSource</code>. DataSource name of the SuperList Default value is empty/<code>undefined</code>
getHeight() on page 134	Getter for property <code>height</code>. The height of the SuperList Default value is <code>100%</code>
getItem(iRow, iCol) on page 134	Gets the value of item.
getMetadataFile() on page 134	Getter for property <code>metadataFile</code>. Metadata file name of the SuperList Default value is empty/<code>undefined</code>

getNumberOfRows() on page 135	Gets number of rows.
getObjectProperty(sName) on page 135	Gets the value of property
getReadRows() on page 136	Getter for property <code>readRows</code>. The default number of rows of dara is retrieved for the SuperList Default value is <code>200</code>
getRow(iRow) on page 136	Gets the values of row
getWidth() on page 137	Getter for property <code>width</code>. The width of the SuperList Default value is <code>100%</code>
insertRow(oRow) on page 137	Inserts a row or appends a row if the input value is -1.
load(sUrl) on page 137	Loads a metafile and creates a superlist object.
refreshData() on page 138	Retrieve data for the SuperList.
reset() on page 138	Resets the data in datatable.
retrieve(oArgs) on page 139	Retrieves data.
setData(sValue, sId) on page 139	Sets data to data table
setDataSource(sDataSource) on page 140	Setter for property <code>dataSource</code>. Default value is empty/<code>undefined</code>
setHeight(sHeight) on page 140	Setter for property <code>height</code>. Default value is <code>100%</code>
setItem(iRow, iCol, oValue) on page 141	Sets the value of a row and column.
setMetadataFile(sMetadataFile) on page 141	Setter for property <code>metadataFile</code>. Default value is empty/<code>undefined</code>
setObjectProperty(sName, sValue) on page 142	Sets value to property.
setReadRows(sReadRows) on page 142	Setter for property <code>readRows</code>. Default value is <code>200</code>
setWidth(sWidth) on page 143	Setter for property <code>width</code>. Default value is <code>100%</code>
sort(sValue) on page 143	Sorts the row.
update() on page 144	Updates changes to data source.

Events

Name	Description
buttonClicked(oControlEvent, oControlEvent.getSource, oControlEvent.getParameters) on page 144	This event is triggered when the button is clicked.
dataTableQuery(oControlEvent, oControlEvent.getSource, oControlEvent.getParameters) on page 144	This event is triggered when the memory data is required.
error(oControlEvent, oControlEvent.getSource, oControlEvent.getParameters) on page 145	This event is triggered when the row focus is changed.
itemChanged(oControlEvent, oControlEvent.getSource, oControlEvent.getParameters) on page 145	This event is triggered when the value of (input, select, checkbox) contol is changed.
rowFocusChanged(oControlEvent, oControlEvent.getSource, oControlEvent.getParameters) on page 146	This event is triggered when the row focus is changed.
updateEnd(oControlEvent, oControlEvent.getSource, oControlEvent.getParameters) on page 146	This event is triggered when the data source update is completed.

Source

SuperList.API.js, line 11 on page 158.

attachButtonClicked([oData], fnFunction, [oListener]) method

Attach event handler <code>fnFunction</code> to the 'buttonClicked' event of this <code>sap.apb.SuperList</code>.

. When called, the context of the event handler (its <code>this</code>) will be bound to <code>oListener<code> if specified otherwise to this <code>sap.apb.SuperList</code>.

itself. This event is triggered when the button is clicked. The args consists of action, row and dataRow.

Syntax

`attachButtonClicked`([*oData*], *fnFunction*, [*oListener*]) {sap.apb.SuperList}

Parameters

Name	Type	Argument	Default	Description
oData	object	(optional)		An application specific payload object, that will be passed to the event handler along with the event object when firing the event.
fnFunction	function			The function to call, when the event occurs.
oListener	object	(optional)	this	Context object to call the event handler with.Defaults to this <code>sap.apb.SuperList</code>. itself.

Returns

<code>this</code> to allow method chaining

Type:

sap.apb.SuperList on page 111

Source

SuperList.API.js, line 166 on page 163.

attachDataTableQuery([oData], fnFunction, [oListener]) method

Attach event handler <code>fnFunction</code> to the 'dataTableQuery' event of this <code>sap.apb.SuperList</code>.

. When called, the context of the event handler (its <code>this</code>) will be bound to <code>oListener<code> if specified otherwise to this <code>sap.apb.SuperList</code>.

itself. This event is triggered when the memory data is required. The args consists of url and retrieveArgs.

Syntax

`attachDataTableQuery`([*oData*], *fnFunction*, [*oListener*]) {sap.apb.SuperList}

Parameters

Name	Type	Argument	Default	Description
oData	object	(optional)		An application specific payload object, that will be passed to the event handler along with the event object when firing the event.
fnFunction	function			The function to call, when the event occurs.
oListener	object	(optional)	this	Context object to call the event handler with.Defaults to this <code>sap.apb.SuperList</code>. itself.

Returns

<code>this</code> to allow method chaining

Type:

sap.apb.SuperList on page 111

Source

SuperList.API.js, line 139 on page 163.

attachError([oData], fnFunction, [oListener]) method

Attach event handler <code>fnFunction</code> to the 'error' event of this <code>sap.apb.SuperList</code>.

. When called, the context of the event handler (its <code>this</code>) will be bound to <code>oListener<code> if specified otherwise to this <code>sap.apb.SuperList</code>.

itself. This event is triggered when the row focus is changed. The args consists of message.

Syntax

`attachError`([*oData*], *fnFunction*, [*oListener*]) {sap.apb.SuperList}

Parameters

Name	Type	Argument	Default	Description
oData	object	(optional)		An application specific payload object, that will be passed to the event handler along with the event object when firing the event.
fnFunction	function			The function to call, when the event occurs.
oListener	object	(optional)	this	Context object to call the event handler with.Defaults to this <code>sap.apb.SuperList</code>. itself.

Returns

<code>this</code> to allow method chaining

Type:

sap.apb.SuperList on page 111

Source
SuperList.API.js, line 184 on page 164.

attachItemChanged([oData], fnFunction, [oListener]) method

Attach event handler <code>fnFunction</code> to the 'itemChanged' event of this <code>sap.apb.SuperList</code>.

. When called, the context of the event handler (its <code>this</code>) will be bound to <code>oListener<code> if specified otherwise to this <code>sap.apb.SuperList</code>.

itself. This event is triggered when the value of (input, select, checkbox) contol is changed. The args consists of row, column & value.

Syntax
`attachItemChanged`([*oData*], *fnFunction*, [*oListener*]) {sap.apb.SuperList}

Parameters

Name	Type	Argument	Default	Description
oData	object	(optional)		An application specific payload object, that will be passed to the event handler along with the event object when firing the event.
fnFunction	function			The function to call, when the event occurs.
oListener	object	(optional)	this	Context object to call the event handler with.Defaults to this <code>sap.apb.SuperList</code>. itself.

Returns
<code>this</code> to allow method chaining

Type:

sap.apb.SuperList on page 111

Source
SuperList.API.js, line 157 on page 163.

attachRowFocusChanged([oData], fnFunction, [oListener]) method

Attach event handler <code>fnFunction</code> to the 'rowFocusChanged' event of this <code>sap.apb.SuperList</code>.

. When called, the context of the event handler (its <code>this</code>) will be bound to <code>oListener<code> if specified otherwise to this <code>sap.apb.SuperList</code>.

itself. This event is triggered when the row focus is changed. The args consists of id & row.

Syntax
`attachRowFocusChanged`([*oData*], *fnFunction*, [*oListener*]) {sap.apb.SuperList}

Parameters

Name	Type	Argument	Default	Description
oData	object	(optional)		An application specific payload object, that will be passed to the event handler along with the event object when firing the event.
fnFunction	function			The function to call, when the event occurs.

oListener	object	(optional)	this	Context object to call the event handler with.Defaults to this <code>sap.apb.SuperList</code>. itself.

Returns
<code>this</code> to allow method chaining

Type:

sap.apb.SuperList on page 111

Source
SuperList.API.js, line 148 on page 163.

attachUpdateEnd([oData], fnFunction, [oListener]) method

Attach event handler <code>fnFunction</code> to the 'updateEnd' event of this <code>sap.apb.SuperList</code>.

. When called, the context of the event handler (its <code>this</code>) will be bound to <code>oListener<code> if specified otherwise to this <code>sap.apb.SuperList</code>.

itself. This event is triggered when the data source update is completed.

Syntax
`attachUpdateEnd`([*oData*], *fnFunction*, [*oListener*]) {sap.apb.SuperList}

Parameters

Name	Type	Argument	Default	Description

oData	object	(optional)		An application specific payload object, that will be passed to the event handler along with the event object when firing the event.
fnFunction	function			The function to call, when the event occurs.
oListener	object	(optional)	this	Context object to call the event handler with.Defaults to this <code>sap.apb.SuperList</code>. itself.

Returns
<code>this</code> to allow method chaining

Type:

sap.apb.SuperList on page 111

Source
SuperList.API.js, line 175 on page 164.

deleteRow(iRow) method

Deletes a row.

Syntax
`deleteRow`(*iRow*) {boolean}

Parameters

Name	Type	Description
iRow	int	row index

Returns

Type:

boolean

Source
SuperList.API.js, line 191 on page 164.

detachButtonClicked(fnFunction, oListener) method

Detach event handler <code>fnFunction</code> from the 'buttonClicked' event of this <code>sap.apb.SuperList</code>.

The passed function and listener object must match the ones used for event registration.

Syntax
`detachButtonClicked`(*fnFunction*, *oListener*) {sap.apb.SuperList}

Parameters

Name	Type	Description
fnFunction	function	The function to call, when the event occurs.
oListener	object	Context object on which the given function had to be called.

Returns
<code>this</code> to allow method chaining

Type:

sap.apb.SuperList on page 111

Source
SuperList.API.js, line 168 on page 164.

detachDataTableQuery(fnFunction, oListener) method

Detach event handler <code>fnFunction</code> from the 'dataTableQuery' event of this <code>sap.apb.SuperList</code>.

The passed function and listener object must match the ones used for event registration.

Syntax

`detachDataTableQuery`(*fnFunction*, *oListener*) {sap.apb.SuperList}

Parameters

Name	Type	Description
fnFunction	function	The function to call, when the event occurs.
oListener	object	Context object on which the given function had to be called.

Returns

<code>this</code> to allow method chaining

Type:

sap.apb.SuperList on page 111

Source

SuperList.API.js, line 141 on page 163.

detachError(fnFunction, oListener) method

Detach event handler <code>fnFunction</code> from the 'error' event of this <code>sap.apb.SuperList</code>.

The passed function and listener object must match the ones used for event registration.

Syntax

`detachError`(*fnFunction*, *oListener*) {sap.apb.SuperList}

Parameters

Name	Type	Description
fnFunction	function	The function to call, when the event occurs.
oListener	object	Context object on which the given function had to be called.

Returns

<code>this</code> to allow method chaining

Type:

sap.apb.SuperList on page 111

Source
SuperList.API.js, line 186 on page 164.

detachItemChanged(fnFunction, oListener) method

Detach event handler <code>fnFunction</code> from the 'itemChanged' event of this <code>sap.apb.SuperList</code>.

The passed function and listener object must match the ones used for event registration.

Syntax
`detachItemChanged`(*fnFunction*, *oListener*) {sap.apb.SuperList}

Parameters

Name	Type	Description
fnFunction	function	The function to call, when the event occurs.
oListener	object	Context object on which the given function had to be called.

Returns
<code>this</code> to allow method chaining

Type:

sap.apb.SuperList on page 111

Source
SuperList.API.js, line 159 on page 163.

detachRowFocusChanged(fnFunction, oListener) method

Detach event handler <code>fnFunction</code> from the 'rowFocusChanged' event of this <code>sap.apb.SuperList</code>.

The passed function and listener object must match the ones used for event registration.

Syntax
`detachRowFocusChanged`(*fnFunction*, *oListener*) {sap.apb.SuperList}

Parameters

Name	Type	Description
fnFunction	function	The function to call, when the event occurs.
oListener	object	Context object on which the given function had to be called.

Returns

<code>this</code> to allow method chaining

Type:

sap.apb.SuperList on page 111

Source

SuperList.API.js, line 150 on page 163.

detachUpdateEnd(fnFunction, oListener) method

Detach event handler <code>fnFunction</code> from the 'updateEnd' event of this <code>sap.apb.SuperList</code>.

The passed function and listener object must match the ones used for event registration.

Syntax

`detachUpdateEnd`(*fnFunction*, *oListener*) {sap.apb.SuperList}

Parameters

Name	Type	Description
fnFunction	function	The function to call, when the event occurs.
oListener	object	Context object on which the given function had to be called.

Returns

<code>this</code> to allow method chaining

Type:

sap.apb.SuperList on page 111

Source
SuperList.API.js, line 177 on page 164.

drillBack() method

Drills back to prior level

Syntax
`drillBack()` {boolean}

Returns

Type:

boolean

Source
SuperList.API.js, line 194 on page 164.

drillDown(iRow) method

Drills down to next level.

Syntax
`drillDown(` *iRow* `)` {boolean}

Parameters

Name	Type	Description
iRow	int	row index

Returns

Type:

boolean

Source
SuperList.API.js, line 245 on page 166.

extend(sClassName, [oClassInfo], [FNMetaImpl]) method

Creates a new subclass of class sap.apb.SuperList with name <code>sClassName</code> and enriches it with the information contained in <code>oClassInfo</code>. <code>oClassInfo</

code> might contain the same kind of informations as described in sap.ui.core.Element.extend Element.extend.

Syntax
<static> `extend`(*sClassName*, [*oClassInfo*], [*FNMetaImpl*]) {function}

Parameters

Name	Type	Argument	Description
sClassName	string		name of the class to be created
oClassInfo	object	(optional)	object literal with informations about the class
FNMetaImpl	function	(optional)	constructor function for the metadata object.If not given, it defaults to sap.ui.core.ElementMetadata.

Returns
the created class / constructor function

Type:

function

Source
SuperList.API.js, line 93 on page 161.

filter(sExpr) method

Filters the rows.

Rows that do not meet the filter criteria are moved to the filtered buffer.

Syntax
`filter`(*sExpr*) {boolean}

Parameters

Name	Type	Description
sExpr	string	Expression of the filter

Returns

Type:

boolean

Source
SuperList.API.js, line 197 on page 165.

fireButtonClicked([mArguments]) method

Fire event buttonClicked to attached listeners.

Syntax
`fireButtonClicked`([*mArguments*]) {sap.apb.SuperList}

Parameters

Name	Type	Argument	Description
mArguments	Map	(optional)	the arguments to pass along with the event.

Returns
<code>this</code> to allow method chaining

Type:

sap.apb.SuperList on page 111

Source
SuperList.API.js, line 170 on page 164.

fireDataTableQuery([mArguments]) method

Fire event dataTableQuery to attached listeners.

Syntax
`fireDataTableQuery`([*mArguments*]) {sap.apb.SuperList}

Parameters

Name	Type	Argument	Description
mArguments	Map	(optional)	the arguments to pass along with the event.

Returns
<code>this</code> to allow method chaining

Type:

sap.apb.SuperList on page 111

Source
SuperList.API.js, line 143 on page 163.

fireError([mArguments]) method

Fire event error to attached listeners.

Syntax
`fireError`([*mArguments*]) {sap.apb.SuperList}

Parameters

Name	Type	Argument	Description
mArguments	Map	(optional)	the arguments to pass along with the event.

Returns
<code>this</code> to allow method chaining

Type:

sap.apb.SuperList on page 111

Source
SuperList.API.js, line 188 on page 164.

fireItemChanged([mArguments]) method

Fire event itemChanged to attached listeners.

Syntax
`fireItemChanged`([*mArguments*]) {sap.apb.SuperList}

Parameters

Name	Type	Argument	Description
mArguments	Map	(optional)	the arguments to pass along with the event.

Returns
<code>this</code> to allow method chaining

Type:

sap.apb.SuperList on page 111

Source
SuperList.API.js, line 161 on page 163.

fireRowFocusChanged([mArguments]) method

Fire event rowFocusChanged to attached listeners.

Syntax
`fireRowFocusChanged`([*mArguments*]) {sap.apb.SuperList}

Parameters

Name	Type	Argument	Description
mArguments	Map	(optional)	the arguments to pass along with the event.

Returns
<code>this</code> to allow method chaining

Type:

sap.apb.SuperList on page 111

Source
SuperList.API.js, line 152 on page 163.

fireUpdateEnd([mArguments]) method

Fire event updateEnd to attached listeners.

Syntax
`fireUpdateEnd`([*mArguments*]) {sap.apb.SuperList}

Parameters

Name	Type	Argument	Description
mArguments	Map	(optional)	the arguments to pass along with the event.

Returns
<code>this</code> to allow method chaining

Type:

sap.apb.SuperList on page 111

Source
SuperList.API.js, line 179 on page 164.

getCurrentLevel() method

Gets current level.

Syntax
`getCurrentLevel()` {int}

Returns

Type:

int

Source
SuperList.API.js, line 200 on page 165.

getDataSource() method

Getter for property <code>dataSource</code>. DataSource name of the SuperList Default value is empty/<code>undefined</code>

Syntax
`getDataSource()` {string}

Returns
the value of property <code>dataSource</code>

Type:

string

Source
SuperList.API.js, line 112 on page 162.

getHeight() method

Getter for property <code>height</code>. The height of the SuperList Default value is <code>100%</code>

Syntax

`getHeight()` {sap.ui.core.CSSSize}

Returns

the value of property <code>height</code>

Type:

sap.ui.core.CSSSize

Source

SuperList.API.js, line 117 on page 162.

getItem(iRow, iCol) method

Gets the value of item.

Syntax

`getItem(` *iRow*, *iCol* `)` {any}

Parameters

Name	Type	Description
iRow	int	row of the item
iCol	int	col of the item

Returns

Type:

any

Source

SuperList.API.js, line 203 on page 165.

getMetadataFile() method

Getter for property <code>metadataFile</code>. Metadata file name of the SuperList Default value is empty/<code>undefined</code>

Syntax
`getMetadataFile()` {sap.ui.core.URI}

Returns
the value of property <code>metadataFile</code>

Type:

sap.ui.core.URI

Source
SuperList.API.js, line 122 on page 162.

getNumberOfRows() method

Gets number of rows.

Syntax
`getNumberOfRows()` {int}

Returns

Type:

int

Source
SuperList.API.js, line 206 on page 165.

getObjectProperty(sName) method

Gets the value of property

Syntax
`getObjectProperty(` *sName* `)` {any}

Parameters

Name	Type	Description
sName	string	Name of the property

Returns

Type:

any

Source
SuperList.API.js, line 236 on page 166.

getReadRows() method

Getter for property <code>readRows</code>. The default number of rows of dara is retrieved for the SuperList Default value is <code>200</code>

Syntax
`getReadRows()` {string}

Returns
the value of property <code>readRows</code>

Type:

string

Source
SuperList.API.js, line 132 on page 162.

getRow(iRow) method

Gets the values of row

Syntax
`getRow(` *iRow* `)` {object}

Parameters

Name	Type	Description
iRow	int	

Returns

Type:

object

Source
SuperList.API.js, line 209 on page 165.

getWidth() method

Getter for property <code>width</code>. The width of the SuperList Default value is <code>100%</code>

Syntax

`getWidth()` {sap.ui.core.CSSSize}

Returns

the value of property <code>width</code>

Type:

sap.ui.core.CSSSize

Source

SuperList.API.js, line 127 on page 162.

insertRow(oRow) method

Inserts a row or appends a row if the input value is -1.

Syntax

`insertRow(` *oRow* `)` {boolean}

Parameters

Name	Type	Description
oRow	object	

Returns

Type:

boolean

Source

SuperList.API.js, line 212 on page 165.

load(sUrl) method

Loads a metafile and creates a superlist object.

Syntax

`load(` *sUrl* `)` {boolean}

Parameters

Name	Type	Description
sUrl	string	The meta data file path to be loaded

Returns

Type:

boolean

Source
SuperList.API.js, line 218 on page 165.

refreshData() method

Retrieve data for the SuperList.

Syntax
`refreshData()` {boolean}

Returns

Type:

boolean

Source
SuperList.API.js, line 215 on page 165.

reset() method

Resets the data in datatable.

Syntax
`reset()` {boolean}

Returns

Type:

boolean

Source
SuperList.API.js, line 221 on page 165.

retrieve(oArgs) method

Retrieves data.

Syntax

`retrieve`(*oArgs*) {boolean}

Parameters

Name	Type	Description
oArgs	object	Retrieve arguments

Returns

Type:

boolean

Source

SuperList.API.js, line 224 on page 165.

setData(sValue, sId) method

Sets data to data table

Syntax

`setData`(*sValue*, *sId*) {boolean}

Parameters

Name	Type	Description
sValue	string	Thle value of the data
sId	string	The id of the data

Returns

Type:

boolean

Source

SuperList.API.js, line 242 on page 166.

setDataSource(sDataSource) method

Setter for property <code>dataSource</code>. Default value is empty/<code>undefined</code>

Syntax

`setDataSource(` *sDataSource* `)` {sap.apb.SuperList}

Parameters

Name	Type	Description
sDataSource	string	new value for property <code>dataSource</code>

Returns

<code>this</code> to allow method chaining

Type:

sap.apb.SuperList on page 111

Source

SuperList.API.js, line 114 on page 162.

setHeight(sHeight) method

Setter for property <code>height</code>. Default value is <code>100%</code>

Syntax

`setHeight(` *sHeight* `)` {sap.apb.SuperList}

Parameters

Name	Type	Description
sHeight	sap.ui.core.CSSSize	new value for property <code>height</code>

Returns

<code>this</code> to allow method chaining

Type:

sap.apb.SuperList on page 111

Source
SuperList.API.js, line 119 on page 162.

setItem(iRow, iCol, oValue) method

Sets the value of a row and column.

Syntax
`setItem`(*iRow*, *iCol*, *oValue*) {boolean}

Parameters

Name	Type	Description
iRow	int	The row of the item
iCol	int	The col of the item
oValue	object	The value to be set

Returns

Type:

boolean

Source
SuperList.API.js, line 230 on page 166.

setMetadataFile(sMetadataFile) method

Setter for property <code>metadataFile</code>. Default value is empty/<code>undefined</code>

Syntax
`setMetadataFile`(*sMetadataFile*) {sap.apb.SuperList}

Parameters

Name	Type	Description
sMetadataFile	sap.ui.core.URI	new value for property <code>metadataFile</code>

Returns
<code>this</code> to allow method chaining

Type:

sap.apb.SuperList on page 111

Source
SuperList.API.js, line 124 on page 162.

setObjectProperty(sName, sValue) method

Sets value to property.

Syntax
`setObjectProperty`(*sName*, *sValue*) {boolean}

Parameters

Name	Type	Description
sName	string	Name of the property
sValue	string	The value to be set

Returns

Type:

boolean

Source
SuperList.API.js, line 239 on page 166.

setReadRows(sReadRows) method

Setter for property <code>readRows</code>. Default value is <code>200</code>

Syntax
`setReadRows`(*sReadRows*) {sap.apb.SuperList}

Parameters

Name	Type	Description
sReadRows	string	new value for property <code>readRows</code>

Returns
<code>this</code> to allow method chaining

Type:

sap.apb.SuperList on page 111

Source
SuperList.API.js, line 134 on page 162.

setWidth(sWidth) method

Setter for property <code>width</code>. Default value is <code>100%</code>

Syntax
`setWidth(` *sWidth* `)` {sap.apb.SuperList}

Parameters

Name	Type	Description
sWidth	sap.ui.core.CSSSize	new value for property <code>width</code>

Returns
<code>this</code> to allow method chaining

Type:

sap.apb.SuperList on page 111

Source
SuperList.API.js, line 129 on page 162.

sort(sValue) method

Sorts the row.

Syntax
`sort(` *sValue* `)` {boolean}

Parameters

Name	Type	Description
sValue	string	sort value

Returns

Type:

boolean

Source
SuperList.API.js, line 227 on page 166.

update() method

Updates changes to data source.

Syntax
`update()` {boolean}

Returns

Type:

boolean

Source
SuperList.API.js, line 233 on page 166.

buttonClicked(oControlEvent, oControlEvent.getSource, oControlEvent.getParameters) event

This event is triggered when the button is clicked.

The args consists of action, row and dataRow.

Parameters

Name	Type	Description
oControlEvent	sap.ui.base.Event	
oControlEvent.getSource	sap.ui.base.EventProvider	
oControlEvent.getParameters	object	

Source
SuperList.API.js, line 164 on page 163.

dataTableQuery(oControlEvent, oControlEvent.getSource, oControlEvent.getParameters) event

This event is triggered when the memory data is required.

The args consists of url and retrieveArgs.

Parameters

Name	Type	Description
oControlEvent	sap.ui.base.Event	
oControlEvent.getSource	sap.ui.base.EventProvider	
oControlEvent.getParameters	object	

Source
SuperList.API.js, line 137 on page 163.

error(oControlEvent, oControlEvent.getSource, oControlEvent.getParameters) event

This event is triggered when the row focus is changed.

The args consists of message.

Parameters

Name	Type	Description
oControlEvent	sap.ui.base.Event	
oControlEvent.getSource	sap.ui.base.EventProvider	
oControlEvent.getParameters	object	

Source
SuperList.API.js, line 182 on page 164.

itemChanged(oControlEvent, oControlEvent.getSource, oControlEvent.getParameters) event

This event is triggered when the value of (input, select, checkbox) contol is changed.

The args consists of row, column & value.

Parameters

Name	Type	Description
oControlEvent	sap.ui.base.Event	
oControlEvent.getSource	sap.ui.base.EventProvider	
oControlEvent.getParameters	object	

Source
SuperList.API.js, line 155 on page 163.

rowFocusChanged(oControlEvent, oControlEvent.getSource, oControlEvent.getParameters) event

This event is triggered when the row focus is changed.

The args consists of id & row.

Parameters

Name	Type	Description
oControlEvent	sap.ui.base.Event	
oControlEvent.getSource	sap.ui.base.EventProvider	
oControlEvent.getParameters	object	

Source
SuperList.API.js, line 146 on page 163.

updateEnd(oControlEvent, oControlEvent.getSource, oControlEvent.getParameters) event

This event is triggered when the data source update is completed.

Parameters

Name	Type	Description
oControlEvent	sap.ui.base.Event	
oControlEvent.getSource	sap.ui.base.EventProvider	
oControlEvent.getParameters	object	

Source
SuperList.API.js, line 173 on page 164.

sap.apb.TabApp class

TabApp is another root element of a UI5 mobile application besides App control.

Users can navigate among the TabPages through tab-button located at the bottom of the screen.

Version: 1.0

Syntax

new `TabApp`([*sId*], [*mSettings*])

Constructor for a new TabApp. Accepts an object literal <code>mSettings</code> that defines initial property values, aggregated and associated objects as well as event handlers. If the name of a setting is ambiguous (e.g. a property has the same name as an event), then the framework assumes property, aggregation, association, event in that order. To override this automatic resolution, one of the prefixes "aggregation:", "association:" or "event:" can be added to the name of the setting (such a prefixed name must be enclosed in single or double quotes). The supported settings are:

- Properties
  - #getTransition transition : sap.apb.TransitionType (default: sap.apb.TransitionType.door)
- Aggregations
- Associations
- Events

In addition, all settings applicable to the base type sap.m.App#constructor sap.m.App can be used as well.

Extends

- sap.m.App

Parameters

Name	Type	Argument	Description
sId	string	(optional)	id for the new control, generated automatically if no id is given
mSettings	object	(optional)	initial settings for the new control

Methods

Name	Description
extend(sClassName, [oClassInfo], [FNMetaImpl]) on page 148	

getTransition() on page 149	Getter for property <code>transition</code>. transition is the transition effect during navigation Default value is <code>door</code>
setTransition(oTransition) on page 149	Setter for property <code>transition</code>. Default value is <code>door</code>

Source
TabApp.API.js, line 11 on page 188.

extend(sClassName, [oClassInfo], [FNMetaImpl]) method

Creates a new subclass of class sap.apb.TabApp with name <code>sClassName</code> and enriches it with the information contained in <code>oClassInfo</code>. <code>oClassInfo</code> might contain the same kind of informations as described in sap.ui.core.Element.extend Element.extend.

Syntax
<static> `extend`(*sClassName*, [*oClassInfo*], [*FNMetaImpl*]) {function}

Parameters

Name	Type	Argument	Description
sClassName	string		name of the class to be created
oClassInfo	object	(optional)	object literal with informations about the class
FNMetaImpl	function	(optional)	constructor function for the metadata object.If not given, it defaults to sap.ui.core.ElementMetadata.

Returns
the created class / constructor function

Type:

function

Source
TabApp.API.js, line 73 on page 190.

getTransition() method

Getter for property <code>transition</code>. transition is the transition effect during navigation Default value is <code>door</code>

Syntax

`getTransition()` {sap.apb.TransitionType}

Returns

the value of property <code>transition</code>

Type:

sap.apb.TransitionType on page 155

Source

TabApp.API.js, line 76 on page 190.

setTransition(oTransition) method

Setter for property <code>transition</code>. Default value is <code>door</code>

Syntax

`setTransition(` *oTransition* `)` {sap.apb.TabApp}

Parameters

Name	Type	Description
oTransition	*sap.apb.TransitionType* on page 155	new value for property <code>transition</code>

Returns

<code>this</code> to allow method chaining

Type:

sap.apb.TabApp on page 146

Source

TabApp.API.js, line 78 on page 190.

sap.apb.TabPage class

A TabPage is a basic container for a tab-based mobile application screen.

Usually one page is displayed at a time and users can navigate to other TabPage through the tab-button located at the bottom of the screen.

Version: 1.0

Syntax

new `TabPage`([*sId*], [*mSettings*])

Constructor for a new TabPage. Accepts an object literal <code>mSettings</code> that defines initial property values, aggregated and associated objects as well as event handlers. If the name of a setting is ambiguous (e.g. a property has the same name as an event), then the framework assumes property, aggregation, association, event in that order. To override this automatic resolution, one of the prefixes "aggregation:", "association:" or "event:" can be added to the name of the setting (such a prefixed name must be enclosed in single or double quotes). The supported settings are:

- Properties
  - #getTabText tabText : string
  - #getTabIcon tabIcon : sap.ui.core.URI
  - #getTabBadge tabBadge : string
- Aggregations
- Associations
- Events

In addition, all settings applicable to the base type sap.m.Page#constructor sap.m.Page can be used as well.

Extends

- sap.m.Page

Parameters

Name	Type	Argument	Description
sId	string	(optional)	id for the new control, generated automatically if no id is given
mSettings	object	(optional)	initial settings for the new control

Methods

Name	Description
extend(sClassName, [oClassInfo], [FNMetaImpl]) on page 151	
getTabBadge() on page 152	Getter for property <code>tabBadge</code>. tabBadge is the badge text to be shown on the tab button Default value is empty/<code>undefined</code>
getTabIcon() on page 152	Getter for property <code>tabIcon</code>. tabIcon is the icon to be shown on the tab button Default value is empty/<code>undefined</code>
getTabText() on page 153	Getter for property <code>tabText</code>. tabText is the text to be shown on the tab button Default value is empty/<code>undefined</code>
setTabBadge(sTabBadge) on page 153	Setter for property <code>tabBadge</code>. Default value is empty/<code>undefined</code>
setTabIcon(sTabIcon) on page 154	Setter for property <code>tabIcon</code>. Default value is empty/<code>undefined</code>
setTabText(sTabText) on page 154	Setter for property <code>tabText</code>. Default value is empty/<code>undefined</code>

Source

TabPage.API.js, line 11 on page 192.

extend(sClassName, [oClassInfo], [FNMetaImpl]) method

Creates a new subclass of class sap.apb.TabPage with name <code>sClassName</code> and enriches it with the information contained in <code>oClassInfo</code>. <code>oClassInfo</code> might contain the same kind of informations as described in sap.ui.core.Element.extend Element.extend.

Syntax

<static> `extend`(*sClassName*, [*oClassInfo*], [*FNMetaImpl*]) {function}

Parameters

Name	Type	Argument	Description

sClassName	string		name of the class to be created
oClassInfo	object	(optional)	object literal with informations about the class
FNMetaImpl	function	(optional)	constructor function for the metadata object.If not given, it defaults to sap.ui.core.ElementMetadata.

Returns
the created class / constructor function

Type:

function

Source
TabPage.API.js, line 74 on page 194.

getTabBadge() method

Getter for property <code>tabBadge</code>. tabBadge is the badge text to be shown on the tab button Default value is empty/<code>undefined</code>

Syntax
`getTabBadge()` {string}

Returns
the value of property <code>tabBadge</code>

Type:

string

Source
TabPage.API.js, line 87 on page 195.

getTabIcon() method

Getter for property <code>tabIcon</code>. tabIcon is the icon to be shown on the tab button Default value is empty/<code>undefined</code>

Syntax
`getTabIcon()` {sap.ui.core.URI}

Returns
the value of property <code>tabIcon</code>

Type:

sap.ui.core.URI

Source
TabPage.API.js, line 82 on page 195.

getTabText() method

Getter for property <code>tabText</code>. tabText is the text to be shown on the tab button Default value is empty/<code>undefined</code>

Syntax
`getTabText()` {string}

Returns
the value of property <code>tabText</code>

Type:

string

Source
TabPage.API.js, line 77 on page 194.

setTabBadge(sTabBadge) method

Setter for property <code>tabBadge</code>. Default value is empty/<code>undefined</code>

Syntax
`setTabBadge(` *sTabBadge* `)` {sap.apb.TabPage}

Parameters

Name	Type	Description
sTabBadge	string	new value for property <code>tabBadge</code>

Returns
<code>this</code> to allow method chaining

Type:

sap.apb.TabPage on page 150

Source
TabPage.API.js, line 89 on page 195.

setTabIcon(sTabIcon) method

Setter for property <code>tabIcon</code>. Default value is empty/<code>undefined</code>

Syntax
`setTabIcon(` *sTabIcon* `)` {sap.apb.TabPage}

Parameters

Name	Type	Description
sTabIcon	sap.ui.core.URI	new value for property <code>tabIcon</code>

Returns
<code>this</code> to allow method chaining

Type:

sap.apb.TabPage on page 150

Source
TabPage.API.js, line 84 on page 195.

setTabText(sTabText) method

Setter for property <code>tabText</code>. Default value is empty/<code>undefined</code>

Syntax
`setTabText(` *sTabText* `)` {sap.apb.TabPage}

Parameters

Name	Type	Description
sTabText	string	new value for property <code>tabText</code>

Returns
<code>this</code> to allow method chaining

Type:

sap.apb.TabPage on page 150

Source
TabPage.API.js, line 79 on page 194.

sap.apb.TransitionType class

Enumeration for TransitionType

Version: 1.0

Syntax
new `TransitionType`()

Since
1.0

Source
TransitionType.ENUM.js, line 17 on page 198.

door member

door

Syntax
<static> `door`

Source
TransitionType.ENUM.js, line 23 on page 198.

fade member

fade

Syntax
<static> `fade`

Source
TransitionType.ENUM.js, line 29 on page 199.

flip member

flip

Syntax
<static> `flip`

Source
TransitionType.ENUM.js, line 35 on page 199.

show member

show

Syntax
<static> `show`

Source
TransitionType.ENUM.js, line 41 on page 199.

slide member

slide

Syntax
<static> `slide`

Source
TransitionType.ENUM.js, line 47 on page 199.

sap.Util class

SAP Util

Version: 1.0

Syntax
new `Util`()

Methods

Name	Description
setStyle([control], [style]) on page 157	Set style for a control

Source
common.js, line 3 on page 199.

setStyle([control], [style]) method

Set style for a control

Syntax
`setStyle`([*control*], [*style*])

Parameters

Name	Type	Argument	Description
control	object	(optional)	The control is to set style
style	string	(optional)	The style is to be set

Source
common.js, line 14 on page 200.

Source code

SuperList.API.js

```
1       /*
-----------------------------------------------------------------------
---------------
2       * Hint: This is a derived (generated) file. Changes should be
done in the underlying
3        * source files only (*.control, *.js) or they will be lost
after the next generation.
4        *
-----------------------------------------------------------------------
--------------- */
5
```

```
6       // Provides control sap.apb.SuperList.
7       jQuery.sap.declare("sap.apb.SuperList");
8       jQuery.sap.require("sap.apb.library");
9       jQuery.sap.require("sap.ui.core.Control");
10
11      /**
12       * Constructor for a new SuperList.
13       *
14       * Accepts an object literal <code>mSettings</code> that
defines initial
15      * property values, aggregated and associated objects as well
as event handlers.
16       *
17      * If the name of a setting is ambiguous (e.g. a property has
the same name as an event),
18       * then the framework assumes property, aggregation,
association, event in that order.
19      * To override this automatic resolution, one of the prefixes
"aggregation:", "association:"
20      * or "event:" can be added to the name of the setting (such a
prefixed name must be
21       * enclosed in single or double quotes).
22       *
23       * The supported settings are:
24       * <ul>
25       * <li>Properties
26       * <ul>
27       * <li>{@link #getDataSource dataSource} : string</li>
28       * <li>height : sap.ui.core.CSSSize (default: '100%')</li>
29       * <li>{@link #getMetadataFile metadataFile} :
sap.ui.core.URI</li>
30       * <li>{@link #getWidth width} : sap.ui.core.CSSSize
(default: '100%')</li>
31       * <li>{@link #getReadRows readRows} : string (default:
'200')</li></ul>
32       * </li>
```

```
33        * <li>Aggregations
34        * <ul></ul>
35        * </li>
36        * <li>Associations
37        * <ul></ul>
38        * </li>
39        * <li>Events
40        * <ul>
41        * <li>{@link sap.apb.SuperList#event:dataTableQuery
dataTableQuery} : fnListenerFunction or [fnListenerFunction,
oListenerObject] or [oData, fnListenerFunction, oListenerObject]</
li>
42        * <li>{@link sap.apb.SuperList#event:rowFocusChanged
rowFocusChanged} : fnListenerFunction or [fnListenerFunction,
oListenerObject] or [oData, fnListenerFunction, oListenerObject]</
li>
43        * <li>{@link sap.apb.SuperList#event:itemChanged
itemChanged} : fnListenerFunction or [fnListenerFunction,
oListenerObject] or [oData, fnListenerFunction, oListenerObject]</
li>
44        * <li>{@link sap.apb.SuperList#event:buttonClicked
buttonClicked} : fnListenerFunction or [fnListenerFunction,
oListenerObject] or [oData, fnListenerFunction, oListenerObject]</
li>
45       * <li>{@link sap.apb.SuperList#event:updateEnd updateEnd} :
fnListenerFunction or [fnListenerFunction, oListenerObject] or
[oData, fnListenerFunction, oListenerObject]</li>
46        * <li>{@link sap.apb.SuperList#event:error error} :
fnListenerFunction or [fnListenerFunction, oListenerObject] or
[oData, fnListenerFunction, oListenerObject]</li></ul>
47        * </li>
48        * </ul>
49
50        *
51        * @param {string} [sId] id for the new control, generated
automatically if no id is given
52        * @param {object} [mSettings] initial settings for the new
control
53        *
54        * @class
```

```
55        * SuperList control
56        * @extends sap.ui.core.Control
57        *
58        * @author SAP AG
59        * @version 1.0
60        *
61        * @constructor
62        * @public
63        * @name sap.apb.SuperList
64        */
65      sap.ui.core.Control.extend("sap.apb.SuperList", { metadata : {
66
67           // ---- object ----
68           publicMethods : [
69               // methods
70             "deleteRow", "drillBack", "filter", "getCurrentLevel", "getItem", "getNumberOfRows", "getRow", "insertRow", "refreshData", "load", "reset", "retrieve", "sort", "setItem", "update", "getObjectProperty", "setObjectProperty", "setData", "drillDown"
71           ],
72
73           // ---- control specific ----
74           library : "sap.apb",
75           properties : {
76               "dataSource" : {type : "string", group : "", defaultValue : null},
77               "height" : {type : "sap.ui.core.CSSSize", group : "Dimension", defaultValue : '100%'},
78             "metadataFile" : {type : "sap.ui.core.URI", group : "", defaultValue : null},
79               "width" : {type : "sap.ui.core.CSSSize", group : "Dimension", defaultValue : '100%'},
80               "readRows" : {type : "string", group : "Misc", defaultValue : '200'}
81           },
```

```
82          events : {
83              "dataTableQuery" : {},
84              "rowFocusChanged" : {},
85              "itemChanged" : {},
86              "buttonClicked" : {},
87              "updateEnd" : {},
88              "error" : {}
89          }
90      }});
91
92
93      /**
94      * Creates a new subclass of class sap.apb.SuperList with name
<code>sClassName</code>
95       * and enriches it with the information contained in
<code>oClassInfo</code>.
96       *
97       * <code>oClassInfo</code> might contain the same kind of
informations as described in {@link sap.ui.core.Element.extend
Element.extend}.
98       *
99       * @param {string} sClassName name of the class to be
created
100      * @param {object} [oClassInfo] object literal with
informations about the class
101      * @param {function} [FNMetaImpl] constructor function for
the metadata object. If not given, it defaults to
sap.ui.core.ElementMetadata.
102      * @return {function} the created class / constructor
function
103      * @public
104      * @static
105      * @name sap.apb.SuperList.extend
106      * @function
107      */
108
```

```
109     sap.apb.SuperList.M_EVENTS =
{'dataTableQuery':'dataTableQuery','rowFocusChanged':'rowFocusChang
ed','itemChanged':'itemChanged','buttonClicked':'buttonClicked','up
dateEnd':'updateEnd','error':'error'};

110

111

112     /**

113      * Getter for property <code>dataSource</code>.

114      * DataSource name of the SuperList

115      *

116      * Default value is empty/<code>undefined</code>

117      *

118      * @return {string} the value of property <code>dataSource</
code>

119      * @public

120      * @name sap.apb.SuperList#getDataSource

121      * @function

122      */

123

124     /**

125      * Setter for property <code>dataSource</code>.

126      *

127      * Default value is empty/<code>undefined</code>

128      *

129      * @param {string} sDataSource  new value for property
<code>dataSource</code>

130      * @return {sap.apb.SuperList} <code>this</code> to allow
method chaining

131      * @public

132      * @name sap.apb.SuperList#setDataSource

133      * @function

134      */

135

136
```

```
137     /**
138      * Getter for property <code>height</code>.
139      * The height of the SuperList
140      *
141      * Default value is <code>100%</code>
142      *
143      * @return {sap.ui.core.CSSSize} the value of property
<code>height</code>
144      * @public
145      * @name sap.apb.SuperList#getHeight
146      * @function
147      */
148
149     /**
150      * Setter for property <code>height</code>.
151      *
152      * Default value is <code>100%</code>
153      *
154      * @param {sap.ui.core.CSSSize} sHeight  new value for
property <code>height</code>
155      * @return {sap.apb.SuperList} <code>this</code> to allow
method chaining
156      * @public
157      * @name sap.apb.SuperList#setHeight
158      * @function
159      */
160
161
162     /**
163      * Getter for property <code>metadataFile</code>.
164      * Metadata file name of the SuperList
165      *
166      * Default value is empty/<code>undefined</code>
```

```
167      *
168      * @return {sap.ui.core.URI} the value of property
<code>metadataFile</code>
169      * @public
170      * @name sap.apb.SuperList#getMetadataFile
171      * @function
172      */
173
174     /**
175      * Setter for property <code>metadataFile</code>.
176      *
177      * Default value is empty/<code>undefined</code>
178      *
179      * @param {sap.ui.core.URI} sMetadataFile  new value for
property <code>metadataFile</code>
180      * @return {sap.apb.SuperList} <code>this</code> to allow
method chaining
181      * @public
182      * @name sap.apb.SuperList#setMetadataFile
183      * @function
184      */
185
186
187     /**
188      * Getter for property <code>width</code>.
189      * The width of the SuperList
190      *
191      * Default value is <code>100%</code>
192      *
193      * @return {sap.ui.core.CSSSize} the value of property
<code>width</code>
194      * @public
195      * @name sap.apb.SuperList#getWidth
196      * @function
```

```
197      */
198
199     /**
200      * Setter for property <code>width</code>.
201      *
202      * Default value is <code>100%</code>
203      *
204     * @param {sap.ui.core.CSSSize} sWidth  new value for property
<code>width</code>
205      * @return {sap.apb.SuperList} <code>this</code> to allow
method chaining
206      * @public
207      * @name sap.apb.SuperList#setWidth
208      * @function
209      */
210
211
212     /**
213      * Getter for property <code>readRows</code>.
214      * The default number of rows of dara is retrieved for the
SuperList
215      *
216      * Default value is <code>200</code>
217      *
218      * @return {string} the value of property <code>readRows</
code>
219      * @public
220      * @name sap.apb.SuperList#getReadRows
221      * @function
222      */
223
224     /**
225      * Setter for property <code>readRows</code>.
226      *
```

```
227      * Default value is <code>200</code>
228      *
229      * @param {string} sReadRows  new value for property
<code>readRows</code>
230      * @return {sap.apb.SuperList} <code>this</code> to allow
method chaining
231      * @public
232      * @name sap.apb.SuperList#setReadRows
233      * @function
234      */
235
236
237     /**
238      * This event is triggered when the memory data is required.
The args consists of url and retrieveArgs.
239      *
240      * @name sap.apb.SuperList#dataTableQuery
241      * @event
242      * @param {sap.ui.base.Event} oControlEvent
243      * @param {sap.ui.base.EventProvider}
oControlEvent.getSource
244      * @param {object} oControlEvent.getParameters
245
246      * @public
247      */
248
249     /**
250      * Attach event handler <code>fnFunction</code> to the
'dataTableQuery' event of this <code>sap.apb.SuperList</code>.<br/
>.
251      * When called, the context of the event handler (its
<code>this</code>) will be bound to <code>oListener<code> if
specified
252      * otherwise to this <code>sap.apb.SuperList</code>.<br/>
itself.
253      *
```

```
254     * This event is triggered when the memory data is required.
The args consists of url and retrieveArgs.

255      *

256      * @param {object}

257     *           [oData] An application specific payload object,
that will be passed to the event handler along with the event object
when firing the event.

258      * @param {function}

259     *           fnFunction The function to call, when the event
occurs.

260      * @param {object}

261     *          [oListener=this] Context object to call the event
handler with. Defaults to this <code>sap.apb.SuperList</code>.<br/>
itself.

262      *

263      * @return {sap.apb.SuperList} <code>this</code> to allow
method chaining

264      * @public

265      * @name sap.apb.SuperList#attachDataTableQuery

266      * @function

267      */

268

269     /**

270      * Detach event handler <code>fnFunction</code> from the
'dataTableQuery' event of this <code>sap.apb.SuperList</code>.<br/>

271      *

272     * The passed function and listener object must match the ones
used for event registration.

273      *

274      * @param {function}

275     *           fnFunction The function to call, when the event
occurs.

276      * @param {object}

277      *            oListener Context object on which the given
function had to be called.

278      * @return {sap.apb.SuperList} <code>this</code> to allow
method chaining
```

```
     * @public
     * @name sap.apb.SuperList#detachDataTableQuery
     * @function
     */

    /**
     * Fire event dataTableQuery to attached listeners.

    * @param {Map} [mArguments] the arguments to pass along with the event.
     * @return {sap.apb.SuperList} <code>this</code> to allow method chaining
     * @protected
     * @name sap.apb.SuperList#fireDataTableQuery
     * @function
     */


    /**
    * This event is triggered when the row focus is changed. The args consists of id & row.
     *
     * @name sap.apb.SuperList#rowFocusChanged
     * @event
     * @param {sap.ui.base.Event} oControlEvent
     * @param {sap.ui.base.EventProvider} oControlEvent.getSource
     * @param {object} oControlEvent.getParameters

     * @public
     */

    /**
```

```
308      * Attach event handler <code>fnFunction</code> to the
'rowFocusChanged' event of this <code>sap.apb.SuperList</code>.<br/
>.

309      * When called, the context of the event handler (its
<code>this</code>) will be bound to <code>oListener<code> if
specified

310      * otherwise to this <code>sap.apb.SuperList</code>.<br/>
itself.

311      *

312     * This event is triggered when the row focus is changed. The
args consists of id & row.

313      *

314      * @param {object}

315     *           [oData] An application specific payload object,
that will be passed to the event handler along with the event object
when firing the event.

316      * @param {function}

317     *           fnFunction The function to call, when the event
occurs.

318      * @param {object}

319     *          [oListener=this] Context object to call the event
handler with. Defaults to this <code>sap.apb.SuperList</code>.<br/>
itself.

320      *

321      * @return {sap.apb.SuperList} <code>this</code> to allow
method chaining

322      * @public

323      * @name sap.apb.SuperList#attachRowFocusChanged

324      * @function

325      */

326

327     /**

328      * Detach event handler <code>fnFunction</code> from the
'rowFocusChanged' event of this <code>sap.apb.SuperList</code>.<br/
>

329      *

330     * The passed function and listener object must match the ones
used for event registration.
```

```
331      *
332      * @param {function}
333     *           fnFunction The function to call, when the event
occurs.
334      * @param {object}
335      *            oListener Context object on which the given
function had to be called.
336      * @return {sap.apb.SuperList} <code>this</code> to allow
method chaining
337      * @public
338      * @name sap.apb.SuperList#detachRowFocusChanged
339      * @function
340      */
341
342     /**
343      * Fire event rowFocusChanged to attached listeners.
344
345     * @param {Map} [mArguments] the arguments to pass along with
the event.
346      * @return {sap.apb.SuperList} <code>this</code> to allow
method chaining
347      * @protected
348      * @name sap.apb.SuperList#fireRowFocusChanged
349      * @function
350      */
351
352
353     /**
354      * This event is triggered when the value of (input, select,
checkbox) contol is changed. The args consists of row, column &
value.
355      *
356      * @name sap.apb.SuperList#itemChanged
357      * @event
358      * @param {sap.ui.base.Event} oControlEvent
```

```
359      * @param {sap.ui.base.EventProvider}
oControlEvent.getSource

360      * @param {object} oControlEvent.getParameters

361

362      * @public

363      */

364

365     /**

366      * Attach event handler <code>fnFunction</code> to the
'itemChanged' event of this <code>sap.apb.SuperList</code>.<br/>.

367      * When called, the context of the event handler (its
<code>this</code>) will be bound to <code>oListener<code> if
specified

368      * otherwise to this <code>sap.apb.SuperList</code>.<br/>
itself.

369      *

370      * This event is triggered when the value of (input, select,
checkbox) contol is changed. The args consists of row, column &
value.

371      *

372      * @param {object}

373      *           [oData] An application specific payload object,
that will be passed to the event handler along with the event object
when firing the event.

374      * @param {function}

375      *           fnFunction The function to call, when the event
occurs.

376      * @param {object}

377     *          [oListener=this] Context object to call the event
handler with. Defaults to this <code>sap.apb.SuperList</code>.<br/>
itself.

378      *

379      * @return {sap.apb.SuperList} <code>this</code> to allow
method chaining

380      * @public

381      * @name sap.apb.SuperList#attachItemChanged

382      * @function
```

```
383      */
384
385     /**
386      * Detach event handler <code>fnFunction</code> from the
'itemChanged' event of this <code>sap.apb.SuperList</code>.<br/>
387      *
388     * The passed function and listener object must match the ones
used for event registration.
389      *
390      * @param {function}
391      *           fnFunction The function to call, when the event
occurs.
392      * @param {object}
393      *            oListener Context object on which the given
function had to be called.
394      * @return {sap.apb.SuperList} <code>this</code> to allow
method chaining
395      * @public
396      * @name sap.apb.SuperList#detachItemChanged
397      * @function
398      */
399
400     /**
401      * Fire event itemChanged to attached listeners.
402
403     * @param {Map} [mArguments] the arguments to pass along with
the event.
404      * @return {sap.apb.SuperList} <code>this</code> to allow
method chaining
405      * @protected
406      * @name sap.apb.SuperList#fireItemChanged
407      * @function
408      */
409
410
```

```
411     /**
412      * This event is triggered when the button is clicked. The
args consists of action, row and dataRow.
413      *
414      * @name sap.apb.SuperList#buttonClicked
415      * @event
416      * @param {sap.ui.base.Event} oControlEvent
417      * @param {sap.ui.base.EventProvider}
oControlEvent.getSource
418      * @param {object} oControlEvent.getParameters
419
420      * @public
421      */
422
423     /**
424      * Attach event handler <code>fnFunction</code> to the
'buttonClicked' event of this <code>sap.apb.SuperList</code>.<br/>.
425      * When called, the context of the event handler (its
<code>this</code>) will be bound to <code>oListener<code> if
specified
426      * otherwise to this <code>sap.apb.SuperList</code>.<br/>
itself.
427      *
428      * This event is triggered when the button is clicked. The
args consists of action, row and dataRow.
429      *
430      * @param {object}
431     *           [oData] An application specific payload object,
that will be passed to the event handler along with the event object
when firing the event.
432      * @param {function}
433     *           fnFunction The function to call, when the event
occurs.
434      * @param {object}
435     *          [oListener=this] Context object to call the event
handler with. Defaults to this <code>sap.apb.SuperList</code>.<br/>
itself.
```

```
436      *
437      * @return {sap.apb.SuperList} <code>this</code> to allow
method chaining
438      * @public
439      * @name sap.apb.SuperList#attachButtonClicked
440      * @function
441      */
442
443     /**
444      * Detach event handler <code>fnFunction</code> from the
'buttonClicked' event of this <code>sap.apb.SuperList</code>.<br/>
445      *
446     * The passed function and listener object must match the ones
used for event registration.
447      *
448      * @param {function}
449     *           fnFunction The function to call, when the event
occurs.
450      * @param {object}
451      *            oListener Context object on which the given
function had to be called.
452      * @return {sap.apb.SuperList} <code>this</code> to allow
method chaining
453      * @public
454      * @name sap.apb.SuperList#detachButtonClicked
455      * @function
456      */
457
458     /**
459      * Fire event buttonClicked to attached listeners.
460
461     * @param {Map} [mArguments] the arguments to pass along with
the event.
462      * @return {sap.apb.SuperList} <code>this</code> to allow
method chaining
```

```
463      * @protected
464      * @name sap.apb.SuperList#fireButtonClicked
465      * @function
466      */
467
468
469     /**
470      * This event is triggered when the data source update is
completed.
471      *
472      * @name sap.apb.SuperList#updateEnd
473      * @event
474      * @param {sap.ui.base.Event} oControlEvent
475      * @param {sap.ui.base.EventProvider}
oControlEvent.getSource
476      * @param {object} oControlEvent.getParameters
477
478      * @public
479      */
480
481     /**
482      * Attach event handler <code>fnFunction</code> to the
'updateEnd' event of this <code>sap.apb.SuperList</code>.<br/>.
483      * When called, the context of the event handler (its
<code>this</code>) will be bound to <code>oListener<code> if
specified
484      * otherwise to this <code>sap.apb.SuperList</code>.<br/>
itself.
485      *
486      * This event is triggered when the data source update is
completed.
487      *
488      * @param {object}
489     *           [oData] An application specific payload object,
that will be passed to the event handler along with the event object
when firing the event.
```

```
490      * @param {function}
491      *           fnFunction The function to call, when the event occurs.
492      * @param {object}
493      *          [oListener=this] Context object to call the event handler with. Defaults to this <code>sap.apb.SuperList</code>.<br/> itself.
494      *
495      * @return {sap.apb.SuperList} <code>this</code> to allow method chaining
496      * @public
497      * @name sap.apb.SuperList#attachUpdateEnd
498      * @function
499      */
500
501     /**
502      * Detach event handler <code>fnFunction</code> from the 'updateEnd' event of this <code>sap.apb.SuperList</code>.<br/>
503      *
504     * The passed function and listener object must match the ones used for event registration.
505      *
506      * @param {function}
507      *           fnFunction The function to call, when the event occurs.
508      * @param {object}
509      *            oListener Context object on which the given function had to be called.
510      * @return {sap.apb.SuperList} <code>this</code> to allow method chaining
511      * @public
512      * @name sap.apb.SuperList#detachUpdateEnd
513      * @function
514      */
515
516     /**
```

```
517      * Fire event updateEnd to attached listeners.
518
519     * @param {Map} [mArguments] the arguments to pass along with
the event.
520      * @return {sap.apb.SuperList} <code>this</code> to allow
method chaining
521      * @protected
522      * @name sap.apb.SuperList#fireUpdateEnd
523      * @function
524      */
525
526
527     /**
528     * This event is triggered when the row focus is changed. The
args consists of message.
529      *
530      * @name sap.apb.SuperList#error
531      * @event
532      * @param {sap.ui.base.Event} oControlEvent
533      * @param {sap.ui.base.EventProvider}
oControlEvent.getSource
534      * @param {object} oControlEvent.getParameters
535
536      * @public
537      */
538
539     /**
540      * Attach event handler <code>fnFunction</code> to the
'error' event of this <code>sap.apb.SuperList</code>.<br/>.
541      * When called, the context of the event handler (its
<code>this</code>) will be bound to <code>oListener<code> if
specified
542      * otherwise to this <code>sap.apb.SuperList</code>.<br/>
itself.
543      *
```

```
544     * This event is triggered when the row focus is changed. The
args consists of message.
545      *
546      * @param {object}
547     *           [oData] An application specific payload object,
that will be passed to the event handler along with the event object
when firing the event.
548      * @param {function}
549     *           fnFunction The function to call, when the event
occurs.
550      * @param {object}
551     *          [oListener=this] Context object to call the event
handler with. Defaults to this <code>sap.apb.SuperList</code>.<br/>
itself.
552      *
553      * @return {sap.apb.SuperList} <code>this</code> to allow
method chaining
554      * @public
555      * @name sap.apb.SuperList#attachError
556      * @function
557      */
558
559     /**
560      * Detach event handler <code>fnFunction</code> from the
'error' event of this <code>sap.apb.SuperList</code>.<br/>
561      *
562     * The passed function and listener object must match the ones
used for event registration.
563      *
564      * @param {function}
565     *           fnFunction The function to call, when the event
occurs.
566      * @param {object}
567      *            oListener Context object on which the given
function had to be called.
568      * @return {sap.apb.SuperList} <code>this</code> to allow
method chaining
```

```
     * @public
     * @name sap.apb.SuperList#detachError
     * @function
     */

    /**
     * Fire event error to attached listeners.

    * @param {Map} [mArguments] the arguments to pass along with the event.
     * @return {sap.apb.SuperList} <code>this</code> to allow method chaining
     * @protected
     * @name sap.apb.SuperList#fireError
     * @function
     */


    /**
     * Deletes a row.
     *
     * @name sap.apb.SuperList.prototype.deleteRow
     * @function
     * @param {int}
     *         iRow
     *         row index

     * @type boolean
     * @public
     */


    /**
```

```
600      * Drills back to prior level
601      *
602      * @name sap.apb.SuperList.prototype.drillBack
603      * @function
604
605      * @type boolean
606      * @public
607      */
608
609
610     /**
611     * Filters the rows. Rows that do not meet the filter criteria
are moved to the filtered buffer.
612      *
613      * @name sap.apb.SuperList.prototype.filter
614      * @function
615      * @param {string}
616      *         sExpr
617      *         Expression of the filter
618
619      * @type boolean
620      * @public
621      */
622
623
624     /**
625      * Gets current level.
626      *
627      * @name sap.apb.SuperList.prototype.getCurrentLevel
628      * @function
629
630      * @type int
```

```
     * @public
     */


    /**
     * Gets the value of item.
     *
     * @name sap.apb.SuperList.prototype.getItem
     * @function
     * @param {int}
     *         iRow
     *         row of the item
     * @param {int}
     *         iCol
     *         col of the item

     * @type any
     * @public
     */


    /**
     * Gets number of rows.
     *
     * @name sap.apb.SuperList.prototype.getNumberOfRows
     * @function

     * @type int
     * @public
     */


```

```
663     /**
664      * Gets the values of row
665      *
666      * @name sap.apb.SuperList.prototype.getRow
667      * @function
668      * @param {int}
669      *         iRow
670      *
671
672      * @type object
673      * @public
674      */
675
676
677     /**
678      * Inserts a row or appends a row if the input value is -1.
679      *
680      * @name sap.apb.SuperList.prototype.insertRow
681      * @function
682      * @param {object}
683      *         oRow
684      *
685
686      * @type boolean
687      * @public
688      */
689
690
691     /**
692      * Retrieve data for the SuperList.
693      *
694      * @name sap.apb.SuperList.prototype.refreshData
```

```
     * @function

     * @type boolean
     * @public
     */


    /**
     * Loads a metafile and creates a superlist object.
     *
     * @name sap.apb.SuperList.prototype.load
     * @function
     * @param {string}
     *         sUrl
     *         The meta data file path to be loaded

     * @type boolean
     * @public
     */


    /**
     * Resets the data in datatable.
     *
     * @name sap.apb.SuperList.prototype.reset
     * @function

     * @type boolean
     * @public
     */


```

```
727     /**
728      * Retrieves data.
729      *
730      * @name sap.apb.SuperList.prototype.retrieve
731      * @function
732      * @param {object}
733      *         oArgs
734      *         Retrieve arguments
735
736      * @type boolean
737      * @public
738      */
739
740
741     /**
742      * Sorts the row.
743      *
744      * @name sap.apb.SuperList.prototype.sort
745      * @function
746      * @param {string}
747      *         sValue
748      *         sort value
749
750      * @type boolean
751      * @public
752      */
753
754
755     /**
756      * Sets the value of a row and column.
757      *
758      * @name sap.apb.SuperList.prototype.setItem
```

```
759      * @function
760      * @param {int}
761      *         iRow
762      *         The row of the item
763      * @param {int}
764      *         iCol
765      *         The col of the item
766      * @param {object}
767      *         oValue
768      *         The value to be set
769
770      * @type boolean
771      * @public
772      */
773
774
775     /**
776      * Updates changes to data source.
777      *
778      * @name sap.apb.SuperList.prototype.update
779      * @function
780
781      * @type boolean
782      * @public
783      */
784
785
786     /**
787      * Gets the value of property
788      *
789      * @name sap.apb.SuperList.prototype.getObjectProperty
790      * @function
```

```
     * @param {string}
     *         sName
     *         Name of the property

     * @type any
     * @public
     */


    /**
     * Sets value to property.
     *
     * @name sap.apb.SuperList.prototype.setObjectProperty
     * @function
     * @param {string}
     *         sName
     *         Name of the property
     * @param {string}
     *         sValue
     *         The value to be set

     * @type boolean
     * @public
     */


    /**
     * Sets data to data table
     *
     * @name sap.apb.SuperList.prototype.setData
     * @function
     * @param {string}
```

```
     *          sValue
     *          Thle value of the data
     * @param {string}
     *          sId
     *          The id of the data

     * @type boolean
     * @public
     */


    /**
     * Drills down to next level.
     *
     * @name sap.apb.SuperList.prototype.drillDown
     * @function
     * @param {int}
     *          iRow
     *          row index

     * @type boolean
     * @public
     */

```

TabApp.API.js

```
      /*
-------------------------------------------------------------------------------
      * Hint: This is a derived (generated) file. Changes should be done in the underlying
       * source files only (*.control, *.js) or they will be lost after the next generation.
```

```
4         *
------------------------------------------------------------------
--------------- */

5

6       // Provides control sap.apb.TabApp.

7       jQuery.sap.declare("sap.apb.TabApp");

8       jQuery.sap.require("sap.apb.library");

9       jQuery.sap.require("sap.m.App");

10

11      /**

12       * Constructor for a new TabApp.

13       *

14       * Accepts an object literal <code>mSettings</code> that
defines initial

15      * property values, aggregated and associated objects as well
as event handlers.

16       *

17      * If the name of a setting is ambiguous (e.g. a property has
the same name as an event),

18       * then the framework assumes property, aggregation,
association, event in that order.

19      * To override this automatic resolution, one of the prefixes
"aggregation:", "association:"

20      * or "event:" can be added to the name of the setting (such a
prefixed name must be

21       * enclosed in single or double quotes).

22       *

23       * The supported settings are:

24       * <ul>

25       * <li>Properties

26       * <ul>

27       * <li>{@link #getTransition transition} :
sap.apb.TransitionType (default: sap.apb.TransitionType.door)</li></
ul>

28       * </li>

29       * <li>Aggregations
```

```
30        * <ul></ul>
31        * </li>
32        * <li>Associations
33        * <ul></ul>
34        * </li>
35        * <li>Events
36        * <ul></ul>
37        * </li>
38        * </ul>
39        *
40        *
41        * In addition, all settings applicable to the base type {@link sap.m.App#constructor sap.m.App}
42        * can be used as well.
43        *
44        * @param {string} [sId] id for the new control, generated automatically if no id is given
45        * @param {object} [mSettings] initial settings for the new control
46        *
47        * @class
48       * TabApp is another root element of a UI5 mobile application besides App control. Users can navigate among the TabPages through tab-button located at the bottom of the screen.
49        * @extends sap.m.App
50        *
51        * @author SAP AG
52        * @version 1.0
53        *
54        * @constructor
55        * @public
56        * @name sap.apb.TabApp
57        */
58       sap.m.App.extend("sap.apb.TabApp", { metadata : {
```

```
59
60          // ---- object ----
61
62          // ---- control specific ----
63          library : "sap.apb",
64          properties : {
65              "transition" : {type : "sap.apb.TransitionType",
group : "", defaultValue : sap.apb.TransitionType.door}
66          },
67          aggregations : {
68              "tabBar" : {type : "sap.m.Bar", multiple : false,
visibility : "hidden"}
69          }
70      }});
71
72
73      /**
74       * Creates a new subclass of class sap.apb.TabApp with name
<code>sClassName</code>
75       * and enriches it with the information contained in
<code>oClassInfo</code>.
76       *
77       * <code>oClassInfo</code> might contain the same kind of
informations as described in {@link sap.ui.core.Element.extend
Element.extend}.
78       *
79       * @param {string} sClassName name of the class to be
created
80       * @param {object} [oClassInfo] object literal with
informations about the class
81      * @param {function} [FNMetaImpl] constructor function for the
metadata object. If not given, it defaults to
sap.ui.core.ElementMetadata.
82       * @return {function} the created class / constructor
function
83       * @public
84       * @static
```

```
85        * @name sap.apb.TabApp.extend
86        * @function
87        */
88
89
90       /**
91        * Getter for property <code>transition</code>.
92        * transition is the transition effect during navigation
93        *
94        * Default value is <code>door</code>
95        *
96        * @return {sap.apb.TransitionType} the value of property
<code>transition</code>
97        * @public
98        * @name sap.apb.TabApp#getTransition
99        * @function
100       */
101
102      /**
103       * Setter for property <code>transition</code>.
104       *
105       * Default value is <code>door</code>
106       *
107      * @param {sap.apb.TransitionType} oTransition  new value for
property <code>transition</code>
108      * @return {sap.apb.TabApp} <code>this</code> to allow method
chaining
109       * @public
110       * @name sap.apb.TabApp#setTransition
111       * @function
112       */
113
114
```

TabPage.API.js

```
1       /*
------------------------------------------------------------------
---------------
2       * Hint: This is a derived (generated) file. Changes should be
done in the underlying
3        * source files only (*.control, *.js) or they will be lost
after the next generation.
4        *
------------------------------------------------------------------
--------------- */
5
6       // Provides control sap.apb.TabPage.
7       jQuery.sap.declare("sap.apb.TabPage");
8       jQuery.sap.require("sap.apb.library");
9       jQuery.sap.require("sap.m.Page");
10
11      /**
12       * Constructor for a new TabPage.
13       *
14       * Accepts an object literal <code>mSettings</code> that
defines initial
15      * property values, aggregated and associated objects as well
as event handlers.
16       *
17      * If the name of a setting is ambiguous (e.g. a property has
the same name as an event),
18       * then the framework assumes property, aggregation,
association, event in that order.
19      * To override this automatic resolution, one of the prefixes
"aggregation:", "association:"
20      * or "event:" can be added to the name of the setting (such a
prefixed name must be
21       * enclosed in single or double quotes).
22       *
23       * The supported settings are:
```

```
24        * <ul>
25        * <li>Properties
26        * <ul>
27        * <li>{@link #getTabText tabText} : string</li>
28        * <li>{@link #getTabIcon tabIcon} : sap.ui.core.URI</li>
29        * <li>{@link #getTabBadge tabBadge} : string</li></ul>
30        * </li>
31        * <li>Aggregations
32        * <ul></ul>
33        * </li>
34        * <li>Associations
35        * <ul></ul>
36        * </li>
37        * <li>Events
38        * <ul></ul>
39        * </li>
40        * </ul>
41        *
42        *
43        * In addition, all settings applicable to the base type
{@link sap.m.Page#constructor sap.m.Page}
44        * can be used as well.
45        *
46        * @param {string} [sId] id for the new control, generated
automatically if no id is given
47        * @param {object} [mSettings] initial settings for the new
control
48        *
49        * @class
50        * A TabPage is a basic container for a tab-based mobile
application screen. Usually one page is displayed at a time and users
can navigate to other TabPage through the tab-button located at the
bottom of the screen.
51        * @extends sap.m.Page
```

```
52        *
53        * @author SAP AG
54        * @version 1.0
55        *
56        * @constructor
57        * @public
58        * @name sap.apb.TabPage
59        */
60       sap.m.Page.extend("sap.apb.TabPage", { metadata : {
61
62           // ---- object ----
63
64           // ---- control specific ----
65           library : "sap.apb",
66           properties : {
67               "tabText" : {type : "string", group : "Misc",
defaultValue : null},
68             "tabIcon" : {type : "sap.ui.core.URI", group : "Misc",
defaultValue : null},
69               "tabBadge" : {type : "string", group : "Misc",
defaultValue : null}
70           }
71       }});
72
73
74       /**
75        * Creates a new subclass of class sap.apb.TabPage with name
<code>sClassName</code>
76        * and enriches it with the information contained in
<code>oClassInfo</code>.
77        *
78        * <code>oClassInfo</code> might contain the same kind of
informations as described in {@link sap.ui.core.Element.extend
Element.extend}.
79        *
```

```
80        * @param {string} sClassName name of the class to be
created

81        * @param {object} [oClassInfo] object literal with
informations about the class

82       * @param {function} [FNMetaImpl] constructor function for the
metadata object. If not given, it defaults to
sap.ui.core.ElementMetadata.

83        * @return {function} the created class / constructor
function

84        * @public

85        * @static

86        * @name sap.apb.TabPage.extend

87        * @function

88        */

89

90

91       /**

92        * Getter for property <code>tabText</code>.

93        * tabText is the text to be shown on the tab button

94        *

95        * Default value is empty/<code>undefined</code>

96        *

97        * @return {string} the value of property <code>tabText</
code>

98        * @public

99        * @name sap.apb.TabPage#getTabText

100       * @function

101       */

102

103      /**

104       * Setter for property <code>tabText</code>.

105       *

106       * Default value is empty/<code>undefined</code>

107       *
```

```
108      * @param {string} sTabText  new value for property
<code>tabText</code>
109      * @return {sap.apb.TabPage} <code>this</code> to allow
method chaining
110      * @public
111      * @name sap.apb.TabPage#setTabText
112      * @function
113      */
114
115
116     /**
117      * Getter for property <code>tabIcon</code>.
118      * tabIcon is the icon to be shown on the tab button
119      *
120      * Default value is empty/<code>undefined</code>
121      *
122      * @return {sap.ui.core.URI} the value of property
<code>tabIcon</code>
123      * @public
124      * @name sap.apb.TabPage#getTabIcon
125      * @function
126      */
127
128     /**
129      * Setter for property <code>tabIcon</code>.
130      *
131      * Default value is empty/<code>undefined</code>
132      *
133      * @param {sap.ui.core.URI} sTabIcon  new value for property
<code>tabIcon</code>
134      * @return {sap.apb.TabPage} <code>this</code> to allow
method chaining
135      * @public
136      * @name sap.apb.TabPage#setTabIcon
```

```
137      * @function
138      */
139
140
141     /**
142      * Getter for property <code>tabBadge</code>.
143      * tabBadge is the badge text to be shown on the tab button
144      *
145      * Default value is empty/<code>undefined</code>
146      *
147      * @return {string} the value of property <code>tabBadge</
code>
148      * @public
149      * @name sap.apb.TabPage#getTabBadge
150      * @function
151      */
152
153     /**
154      * Setter for property <code>tabBadge</code>.
155      *
156      * Default value is empty/<code>undefined</code>
157      *
158      * @param {string} sTabBadge  new value for property
<code>tabBadge</code>
159      * @return {sap.apb.TabPage} <code>this</code> to allow
method chaining
160      * @public
161      * @name sap.apb.TabPage#setTabBadge
162      * @function
163      */
164
165
```

TransitionType.ENUM.js

```
/*
-----------------------------------------------------------------------------
* Hint: This is a derived (generated) file. Changes should be done in the underlying
* source files only (*.type, *.js) or they will be lost after the next generation.
 *
------------------------------------------------------------------------------ */

// Provides enumeration sap.apb.TransitionType.
jQuery.sap.declare("sap.apb.TransitionType");

/**
 * @class Enumeration for TransitionType
 *
 * @version 1.0
 * @static
 * @public
 * @since 1.0
 */
sap.apb.TransitionType = {

    /**
     * door
     * @public
     */
    door : "door",

    /**
     * fade
```

```
27           * @public
28           */
29          fade : "fade",
30
31          /**
32           * flip
33           * @public
34           */
35          flip : "flip",
36
37          /**
38           * show
39           * @public
40           */
41          show : "show",
42
43          /**
44           * slide
45           * @public
46           */
47          slide : "slide"
48
49        };
50
51
```

common.js

```
1       if (typeof(SAP) == 'undefined') SAP = {};
2
3       /**
4        * @class
5        * SAP Util
```

```
6         *
7         * @author SAP AG
8         * @version 1.0
9         *
10        * @public
11        * @name sap.Util
12        */
13       SAP.Util = {
14       /**
15        * Set style for a control
16        *
17        * @param {object} [control] The control is to set style
18        * @param {string} [style] The style is to be set
19        * @public
20        * @static
21        * @name sap.Util#setStyle
22         * @function
23        */
24           setStyle: function(control, style) {
25               if(control instanceof sap.ui.core.Control) {
26                   control.data("style", style);
27               }
28           }
29       };
30
31       /**
32        * @class
33        * SAP DataSources APIs
34        *
35        * @author SAP AG
36        * @version 1.0
37        *
```

```
38       * @public
39       * @name sap.DataSourceAPI
40       */
41      SAP.DataSourceAPI = {
42      /**
43       * Set the application logon context for accesing oData
servicesfrom SMP server. This context is from SMP Kapsel logon
manager.
44       *
45       * @param {string} [smpLogonApplicationContext] Application
login context. This context should be the returned result from the
Kapsel logon manager.
46       * @public
47       * @static
48       * @name sap.DataSourceAPI#setLogonApplicationContext
49       * @function
50       */
51      setLogonApplicationContext: function
(smpLogonApplicationContext) {
52      },
53
54      /**
55       * Change the specified datasourceâ€™s user name, password,
and root URL information
56       *
57      * @param {string} [dataSourceName] The datasource's name for
example "ODataService.EmployeeDB", it canâ€™t be the table source
name
58       * @param {string} [user] The new user name ,if you donâ€™t
want to change the user name, you can set it to undefined
59      * @param {string} [password] The new password, if you donâ€™t
want to change the password, you can set it to undefined
60      * @param {string} [url] The new root URL for the data source,
if you donâ€™t want to change the root URL, you can set it to
undefined
61       * @public
62       * @static
```

```
63        * @name sap.DataSourceAPI#setDataSourceInfo
64        * @function
65        */
66           setDataSourceInfo: function(dataSourceName, user,
password, url) {
67           },
68
69       /**
70        * Change the current SMP Server Profile
71        *
72       * @param {string} [SMPServer]  The new SMP Server to use. If
you donâ€™t want to change it, you can set it to undefined
73        * @param {string} [SMPApplicationID] The new SMP
ApplicationID. If you donâ€™t want to change it, you can set it to
undefined
74        * @param {string} [SMPApplicationTag] The new SMP
ApplicationTag. If you donâ€™t want to change it, you can set it to
undefined
75        * @param {bool} [allowAnonymous] You can set it to true if
the server allows anonymous access, otherwise, set it false. If you
donâ€™t want to change it, you can set it to undefined
76        * @param {string} [user] The new user name. If you donâ€™t
want to change it, you can set it to undefined
77       * @param {string} [password] The new password. If you donâ€™t
want to change it, you can set it to undefined
78       * @param {string} [connectionID] The the conection ID to be
used. If you want to clear the old connection ID, you can set it to
empty string ("").
79        * @public
80        * @static
81        * @name sap.DataSourceAPI#setSMPServerProfile
82        * @function
83        */
84           setSMPServerProfile :
function(SMPServer,SMPApplicationID,SMPApplicationTag,allowAnonymou
s,user,password) {
85           },
86
```

```
87      /**
88       * Encrypt the specified password and return the encrypted
password
89       *
90      * @param {string} [password] The password that is going to be
encrypted
91       * @public
92       * @static
93       * @name sap.DataSourceAPI#encryptPassword
94       * @function
95       */
96          encryptPassword: function(password) {
97          },
98
99      /**
100      * Decrypt the specified password and return the decrypted
password
101      *
102     * @param {string} [password] The password that is going to be
decrypted
103      * @public
104      * @static
105      * @name sap.DataSourceAPI#decryptPassword
106      * @function
107      */
108         decryptPassword: function(password) {
109         }
110     };
111
```

makit/Chart.API.js

```
1       /*
-----------------------------------------------------------------------
---------------
```

```
2       * Hint: This is a derived (generated) file. Changes should be
done in the underlying

3        * source files only (*.control, *.js) or they will be lost
after the next generation.

4        *
------------------------------------------------------------------
--------------- */

5

6       // Provides control sap.apb.makit.Chart.

7       jQuery.sap.declare("sap.apb.makit.Chart");

8       jQuery.sap.require("sap.apb.makit.library");

9       jQuery.sap.require("sap.ui.core.Control");

10

11      /**

12       * Constructor for a new Chart.

13       *

14       * Accepts an object literal <code>mSettings</code> that
defines initial

15      * property values, aggregated and associated objects as well
as event handlers.

16       *

17      * If the name of a setting is ambiguous (e.g. a property has
the same name as an event),

18       * then the framework assumes property, aggregation,
association, event in that order.

19      * To override this automatic resolution, one of the prefixes
"aggregation:", "association:"

20      * or "event:" can be added to the name of the setting (such a
prefixed name must be

21       * enclosed in single or double quotes).

22       *

23       * The supported settings are:

24       * <ul>

25       * <li>Properties

26       * <ul>

27       * <li>{@link #getType type} : sap.apb.makit.ChartType
(default: sap.apb.makit.ChartType.Column)</li>
```

```
28        * <li>{@link #getCategorySortOrder categorySortOrder} :
sap.apb.makit.SortOrder (default: sap.apb.makit.SortOrder.None)</
li>

29        * <li>{@link #getDataSource dataSource} : string</li>

30        * <li>height : sap.ui.core.CSSSize (default: '100%')</li>

31        * <li>{@link #getLegendPosition legendPosition} :
sap.apb.makit.LegendPosition (default:
sap.apb.makit.LegendPosition.None)</li>

32        * <li>{@link #getLineThickness lineThickness} : float
(default: 1)</li>

33        * <li>{@link #getMaxSliceCount maxSliceCount} : int
(default: 12)</li>

34        * <li>{@link #getMetadataFile metadataFile} :
sap.ui.core.URI</li>

35        * <li>{@link #getShowRangeSelector showRangeSelector} :
boolean (default: true)</li>

36        * <li>{@link #getShowTableView showTableView} : boolean
(default: false)</li>

37        * <li>{@link #getShowTableValue showTableValue} : boolean
(default: true)</li>

38        * <li>{@link #getShowToolbar showToolbar} : boolean
(default: true)</li>

39        * <li>{@link #getWidth width} : sap.ui.core.CSSSize
(default: '100%')</li></ul>

40        * </li>

41        * <li>Aggregations

42        * <ul></ul>

43        * </li>

44        * <li>Associations

45        * <ul></ul>

46        * </li>

47        * <li>Events

48        * <ul>

49        * <li>{@link sap.apb.makit.Chart#event:doubletap
doubletap} : fnListenerFunction or [fnListenerFunction,
oListenerObject] or [oData, fnListenerFunction, oListenerObject]</
li>
```

```
50        * <li>{@link sap.apb.makit.Chart#event:tap tap} :
fnListenerFunction or [fnListenerFunction, oListenerObject] or
[oData, fnListenerFunction, oListenerObject]</li></ul>

51        * </li>

52        * </ul>

53

54        *

55        * @param {string} [sId] id for the new control, generated
automatically if no id is given

56        * @param {object} [mSettings] initial settings for the new
control

57        *

58        * @class

59        * MAKit Chart

60        * @extends sap.ui.core.Control

61        *

62        * @author SAP AG

63        * @version 1.0

64        *

65        * @constructor

66        * @public

67        * @name sap.apb.makit.Chart

68        */

69       sap.ui.core.Control.extend("sap.apb.makit.Chart",
{ metadata : {

70

71           // ---- object ----

72           publicMethods : [

73               // methods

74               "getSelectedCategory", "getSelectedSeries",
"getNumberOfCategories", "getSelectedCategoryGroup", "refreshData"

75           ],

76

77           // ---- control specific ----
```

```
78          library : "sap.apb.makit",

79          properties : {

80              "type" : {type : "sap.apb.makit.ChartType", group :
"Appearance", defaultValue : sap.apb.makit.ChartType.Column},

81              "categorySortOrder" : {type :
"sap.apb.makit.SortOrder", group : "Misc", defaultValue :
sap.apb.makit.SortOrder.None},

82              "dataSource" : {type : "string", group : "",
defaultValue : null},

83              "height" : {type : "sap.ui.core.CSSSize", group :
"Dimension", defaultValue : '100%'},

84              "legendPosition" : {type :
"sap.apb.makit.LegendPosition", group : "Appearance", defaultValue :
sap.apb.makit.LegendPosition.None},

85              "lineThickness" : {type : "float", group :
"Appearance", defaultValue : 1},

86              "maxSliceCount" : {type : "int", group : "Misc",
defaultValue : 12},

87            "metadataFile" : {type : "sap.ui.core.URI", group : "",
defaultValue : null},

88              "showRangeSelector" : {type : "boolean", group :
"Appearance", defaultValue : true},

89              "showTableView" : {type : "boolean", group :
"Appearance", defaultValue : false},

90              "showTableValue" : {type : "boolean", group :
"Appearance", defaultValue : true},

91              "showToolbar" : {type : "boolean", group :
"Appearance", defaultValue : true},

92              "width" : {type : "sap.ui.core.CSSSize", group :
"Dimension", defaultValue : '100%'}

93          },

94          events : {

95              "doubletap" : {},

96              "tap" : {}

97          }

98      }});

99

100
```

```
101     /**
102      * Creates a new subclass of class sap.apb.makit.Chart with
name <code>sClassName</code>
103      * and enriches it with the information contained in
<code>oClassInfo</code>.
104      *
105      * <code>oClassInfo</code> might contain the same kind of
informations as described in {@link sap.ui.core.Element.extend
Element.extend}.
106      *
107      * @param {string} sClassName name of the class to be
created
108      * @param {object} [oClassInfo] object literal with
informations about the class
109      * @param {function} [FNMetaImpl] constructor function for
the metadata object. If not given, it defaults to
sap.ui.core.ElementMetadata.
110      * @return {function} the created class / constructor
function
111      * @public
112      * @static
113      * @name sap.apb.makit.Chart.extend
114      * @function
115      */
116
117     sap.apb.makit.Chart.M_EVENTS =
{'doubletap':'doubletap','tap':'tap'};
118
119
120     /**
121      * Getter for property <code>type</code>.
122      * Chart type.
123      *
124      * Default value is <code>Column</code>
125      *
126      * @return {sap.apb.makit.ChartType} the value of property
<code>type</code>
```

```
127      * @public
128      * @name sap.apb.makit.Chart#getType
129      * @function
130      */
131
132     /**
133      * Setter for property <code>type</code>.
134      *
135      * Default value is <code>Column</code>
136      *
137      * @param {sap.apb.makit.ChartType} oType  new value for
property <code>type</code>
138      * @return {sap.apb.makit.Chart} <code>this</code> to allow
method chaining
139      * @public
140      * @name sap.apb.makit.Chart#setType
141      * @function
142      */
143
144
145     /**
146      * Getter for property <code>categorySortOrder</code>.
147     * Sort order for category. If None, the category's column is
expected to be pre-sorted.
148      *
149      * Default value is <code>None</code>
150      *
151      * @return {sap.apb.makit.SortOrder} the value of property
<code>categorySortOrder</code>
152      * @public
153      * @name sap.apb.makit.Chart#getCategorySortOrder
154      * @function
155      */
156
```

```
157     /**
158      * Setter for property <code>categorySortOrder</code>.
159      *
160      * Default value is <code>None</code>
161      *
162      * @param {sap.apb.makit.SortOrder} oCategorySortOrder  new
value for property <code>categorySortOrder</code>
163      * @return {sap.apb.makit.Chart} <code>this</code> to allow
method chaining
164      * @public
165      * @name sap.apb.makit.Chart#setCategorySortOrder
166      * @function
167      */
168
169
170     /**
171      * Getter for property <code>dataSource</code>.
172      * dataSource name of the chart.
173      *
174      * Default value is empty/<code>undefined</code>
175      *
176      * @return {string} the value of property <code>dataSource</
code>
177      * @public
178      * @name sap.apb.makit.Chart#getDataSource
179      * @function
180      */
181
182     /**
183      * Setter for property <code>dataSource</code>.
184      *
185      * Default value is empty/<code>undefined</code>
186      *
```

```
187      * @param {string} sDataSource  new value for property
<code>dataSource</code>
188      * @return {sap.apb.makit.Chart} <code>this</code> to allow
method chaining
189      * @public
190      * @name sap.apb.makit.Chart#setDataSource
191      * @function
192      */
193
194
195     /**
196      * Getter for property <code>height</code>.
197      * The height of the Chart.
198      *
199      * Default value is <code>100%</code>
200      *
201      * @return {sap.ui.core.CSSSize} the value of property
<code>height</code>
202      * @public
203      * @name sap.apb.makit.Chart#getHeight
204      * @function
205      */
206
207     /**
208      * Setter for property <code>height</code>.
209      *
210      * Default value is <code>100%</code>
211      *
212      * @param {sap.ui.core.CSSSize} sHeight  new value for
property <code>height</code>
213      * @return {sap.apb.makit.Chart} <code>this</code> to allow
method chaining
214      * @public
215      * @name sap.apb.makit.Chart#setHeight
```

```
216      * @function
217      */
218
219
220     /**
221      * Getter for property <code>legendPosition</code>.
222     * Legend position all chart types except Bar chart. Bar chart
only support Bottom position.
223      *
224      * Default value is <code>None</code>
225      *
226      * @return {sap.apb.makit.LegendPosition} the value of
property <code>legendPosition</code>
227      * @public
228      * @name sap.apb.makit.Chart#getLegendPosition
229      * @function
230      */
231
232     /**
233      * Setter for property <code>legendPosition</code>.
234      *
235      * Default value is <code>None</code>
236      *
237     * @param {sap.apb.makit.LegendPosition} oLegendPosition  new
value for property <code>legendPosition</code>
238      * @return {sap.apb.makit.Chart} <code>this</code> to allow
method chaining
239      * @public
240      * @name sap.apb.makit.Chart#setLegendPosition
241      * @function
242      */
243
244
245     /**
```

```
     * Getter for property <code>lineThickness</code>.
    * Specify the line thickness of the line graph. Only applies to Line chart type.
     *
     * Default value is <code>1</code>
     *
     * @return {float} the value of property <code>lineThickness</code>
     * @public
     * @name sap.apb.makit.Chart#getLineThickness
     * @function
     */

    /**
     * Setter for property <code>lineThickness</code>.
     *
     * Default value is <code>1</code>
     *
     * @param {float} fLineThickness  new value for property <code>lineThickness</code>
     * @return {sap.apb.makit.Chart} <code>this</code> to allow method chaining
     * @public
     * @name sap.apb.makit.Chart#setLineThickness
     * @function
     */


    /**
     * Getter for property <code>maxSliceCount</code>.
     * Set the maximum number of slices in a Pie/Donut chart. If exceeding the specified value, the rest will be categorised into a single slice. Only applies to Pie/Donut.
     *
```

```
     * Default value is <code>12</code>
     *
     * @return {int} the value of property <code>maxSliceCount</code>
     * @public
     * @name sap.apb.makit.Chart#getMaxSliceCount
     * @function
     */

    /**
     * Setter for property <code>maxSliceCount</code>.
     *
     * Default value is <code>12</code>
     *
     * @param {int} iMaxSliceCount  new value for property <code>maxSliceCount</code>
     * @return {sap.apb.makit.Chart} <code>this</code> to allow method chaining
     * @public
     * @name sap.apb.makit.Chart#setMaxSliceCount
     * @function
     */


    /**
     * Getter for property <code>metadataFile</code>.
     * Metadata file URI that is assigned to this chart. Metadata is mandatory for a chart to be rendered because it contains the chart's details.
     *
     * Default value is empty/<code>undefined</code>
     *
     * @return {sap.ui.core.URI} the value of property <code>metadataFile</code>
```

```
302      * @public
303      * @name sap.apb.makit.Chart#getMetadataFile
304      * @function
305      */
306
307     /**
308      * Setter for property <code>metadataFile</code>.
309      *
310      * Default value is empty/<code>undefined</code>
311      *
312      * @param {sap.ui.core.URI} sMetadataFile  new value for
property <code>metadataFile</code>
313      * @return {sap.apb.makit.Chart} <code>this</code> to allow
method chaining
314      * @public
315      * @name sap.apb.makit.Chart#setMetadataFile
316      * @function
317      */
318
319
320     /**
321      * Getter for property <code>showRangeSelector</code>.
322      * Specify whether the range selector should be visible.
323      *
324      * Default value is <code>true</code>
325      *
326      * @return {boolean} the value of property
<code>showRangeSelector</code>
327      * @public
328      * @name sap.apb.makit.Chart#getShowRangeSelector
329      * @function
330      */
331
```

```
332     /**
333      * Setter for property <code>showRangeSelector</code>.
334      *
335      * Default value is <code>true</code>
336      *
337     * @param {boolean} bShowRangeSelector  new value for property
<code>showRangeSelector</code>
338      * @return {sap.apb.makit.Chart} <code>this</code> to allow
method chaining
339      * @public
340      * @name sap.apb.makit.Chart#setShowRangeSelector
341      * @function
342      */
343
344
345     /**
346      * Getter for property <code>showTableView</code>.
347      * Toggle to display table view.
348      *
349      * Default value is <code>false</code>
350      *
351      * @return {boolean} the value of property
<code>showTableView</code>
352      * @public
353      * @name sap.apb.makit.Chart#getShowTableView
354      * @function
355      */
356
357     /**
358      * Setter for property <code>showTableView</code>.
359      *
360      * Default value is <code>false</code>
361      *
```

```
362      * @param {boolean} bShowTableView  new value for property
<code>showTableView</code>

363      * @return {sap.apb.makit.Chart} <code>this</code> to allow
method chaining

364      * @public

365      * @name sap.apb.makit.Chart#setShowTableView

366      * @function

367      */

368

369

370     /**

371      * Getter for property <code>showTableValue</code>.

372      * Toggle to display the table value on a Bar chart. Only
applies to Bar chart.

373      *

374      * Default value is <code>true</code>

375      *

376      * @return {boolean} the value of property
<code>showTableValue</code>

377      * @public

378      * @name sap.apb.makit.Chart#getShowTableValue

379      * @function

380      */

381

382     /**

383      * Setter for property <code>showTableValue</code>.

384      *

385      * Default value is <code>true</code>

386      *

387      * @param {boolean} bShowTableValue  new value for property
<code>showTableValue</code>

388      * @return {sap.apb.makit.Chart} <code>this</code> to allow
method chaining

389      * @public
```

```
390      * @name sap.apb.makit.Chart#setShowTableValue
391      * @function
392      */
393
394
395     /**
396      * Getter for property <code>showToolbar</code>.
397      * Show or hide the chart's toolbar.
398      *
399      * Default value is <code>true</code>
400      *
401      * @return {boolean} the value of property
<code>showToolbar</code>
402      * @public
403      * @name sap.apb.makit.Chart#getShowToolbar
404      * @function
405      */
406
407     /**
408      * Setter for property <code>showToolbar</code>.
409      *
410      * Default value is <code>true</code>
411      *
412      * @param {boolean} bShowToolbar  new value for property
<code>showToolbar</code>
413      * @return {sap.apb.makit.Chart} <code>this</code> to allow
method chaining
414      * @public
415      * @name sap.apb.makit.Chart#setShowToolbar
416      * @function
417      */
418
419
```

```
420     /**
421      * Getter for property <code>width</code>.
422      * The width of the Chart.
423      *
424      * Default value is <code>100%</code>
425      *
426      * @return {sap.ui.core.CSSSize} the value of property
<code>width</code>
427      * @public
428      * @name sap.apb.makit.Chart#getWidth
429      * @function
430      */
431
432     /**
433      * Setter for property <code>width</code>.
434      *
435      * Default value is <code>100%</code>
436      *
437     * @param {sap.ui.core.CSSSize} sWidth  new value for property
<code>width</code>
438      * @return {sap.apb.makit.Chart} <code>this</code> to allow
method chaining
439      * @public
440      * @name sap.apb.makit.Chart#setWidth
441      * @function
442      */
443
444
445     /**
446      * Double tap event on chart.
447      *
448      * @name sap.apb.makit.Chart#doubletap
449      * @event
```

```
450      * @param {sap.ui.base.Event} oControlEvent

451      * @param {sap.ui.base.EventProvider}
oControlEvent.getSource

452      * @param {object} oControlEvent.getParameters

453

454      * @public

455      */

456

457     /**

458      * Attach event handler <code>fnFunction</code> to the
'doubletap' event of this <code>sap.apb.makit.Chart</code>.<br/>.

459      * When called, the context of the event handler (its
<code>this</code>) will be bound to <code>oListener<code> if
specified

460      * otherwise to this <code>sap.apb.makit.Chart</code>.<br/>
itself.

461      *

462      * Double tap event on chart.

463      *

464      * @param {object}

465     *           [oData] An application specific payload object,
that will be passed to the event handler along with the event object
when firing the event.

466      * @param {function}

467     *           fnFunction The function to call, when the event
occurs.

468      * @param {object}

469     *          [oListener=this] Context object to call the event
handler with. Defaults to this <code>sap.apb.makit.Chart</code>.<br/
> itself.

470      *

471      * @return {sap.apb.makit.Chart} <code>this</code> to allow
method chaining

472      * @public

473      * @name sap.apb.makit.Chart#attachDoubletap

474      * @function

475      */
```

```
476
477     /**
478      * Detach event handler <code>fnFunction</code> from the
'doubletap' event of this <code>sap.apb.makit.Chart</code>.<br/>
479      *
480     * The passed function and listener object must match the ones
used for event registration.
481      *
482      * @param {function}
483     *           fnFunction The function to call, when the event
occurs.
484      * @param {object}
485      *            oListener Context object on which the given
function had to be called.
486      * @return {sap.apb.makit.Chart} <code>this</code> to allow
method chaining
487      * @public
488      * @name sap.apb.makit.Chart#detachDoubletap
489      * @function
490      */
491
492     /**
493      * Fire event doubletap to attached listeners.
494
495     * @param {Map} [mArguments] the arguments to pass along with
the event.
496      * @return {sap.apb.makit.Chart} <code>this</code> to allow
method chaining
497      * @protected
498      * @name sap.apb.makit.Chart#fireDoubletap
499      * @function
500      */
501
502
503     /**
```

```
504      * Single tap event on the chart.
505      *
506      * @name sap.apb.makit.Chart#tap
507      * @event
508      * @param {sap.ui.base.Event} oControlEvent
509      * @param {sap.ui.base.EventProvider}
oControlEvent.getSource
510      * @param {object} oControlEvent.getParameters
511
512      * @public
513      */
514
515     /**
516      * Attach event handler <code>fnFunction</code> to the 'tap'
event of this <code>sap.apb.makit.Chart</code>.<br/>.
517      * When called, the context of the event handler (its
<code>this</code>) will be bound to <code>oListener<code> if
specified
518      * otherwise to this <code>sap.apb.makit.Chart</code>.<br/>
itself.
519      *
520      * Single tap event on the chart.
521      *
522      * @param {object}
523     *           [oData] An application specific payload object,
that will be passed to the event handler along with the event object
when firing the event.
524      * @param {function}
525     *           fnFunction The function to call, when the event
occurs.
526      * @param {object}
527     *          [oListener=this] Context object to call the event
handler with. Defaults to this <code>sap.apb.makit.Chart</code>.<br/
> itself.
528      *
529      * @return {sap.apb.makit.Chart} <code>this</code> to allow
method chaining
```

```
530      * @public
531      * @name sap.apb.makit.Chart#attachTap
532      * @function
533      */
534
535     /**
536      * Detach event handler <code>fnFunction</code> from the
'tap' event of this <code>sap.apb.makit.Chart</code>.<br/>
537      *
538     * The passed function and listener object must match the ones
used for event registration.
539      *
540      * @param {function}
541     *           fnFunction The function to call, when the event
occurs.
542      * @param {object}
543      *            oListener Context object on which the given
function had to be called.
544      * @return {sap.apb.makit.Chart} <code>this</code> to allow
method chaining
545      * @public
546      * @name sap.apb.makit.Chart#detachTap
547      * @function
548      */
549
550     /**
551      * Fire event tap to attached listeners.
552
553     * @param {Map} [mArguments] the arguments to pass along with
the event.
554      * @return {sap.apb.makit.Chart} <code>this</code> to allow
method chaining
555      * @protected
556      * @name sap.apb.makit.Chart#fireTap
557      * @function
```

```
     */


    /**
     * Get the value of the currently highlighted category.
     *
     * @name sap.apb.makit.Chart.prototype.getSelectedCategory
     * @function

     * @type string
     * @public
     */


    /**
     * Get the value of the currently highlighted series.
     *
     * @name sap.apb.makit.Chart.prototype.getSelectedSeries
     * @function

     * @type string
     * @public
     */


    /**
     * Get the number of distinct category values.
     *
     * @name sap.apb.makit.Chart.prototype.getNumberOfCategories
     * @function

```

```
     * @type int
     * @public
     */


    /**
     * Re-retrieve data from the datasource and re-render chart.
     *
     * @name sap.apb.makit.Chart.prototype.refreshData
     * @function

     * @type void
     * @public
     */


```

makit/ChartType.ENUM.js

```
      /*
      * Hint: This is a derived (generated) file. Changes should be done in the underlying
      * source files only (*.type, *.js) or they will be lost after the next generation.
       *
       */

      // Provides enumeration sap.apb.makit.ChartType.
      jQuery.sap.declare("sap.apb.makit.ChartType");

      /**
       * @class Enumeration for chart type
```

```
11        *
12        * @version 1.0
13        * @static
14        * @public
15        * @since 1.0
16        */
17       sap.apb.makit.ChartType = {
18
19           /**
20            * Column chart
21            * @public
22            */
23           Column : "Column",
24
25           /**
26            * Line chart
27            * @public
28            */
29           Line : "Line",
30
31           /**
32            * Bubble chart
33            * @public
34            */
35           Bubble : "Bubble",
36
37           /**
38            * Horizontal table bar chart
39            * @public
40            */
41           Bar : "Bar",
42
```

```
43          /**
44           * Pie chart
45           * @public
46           */
47          Pie : "Pie"
48
49        };
50
51
```

makit/LegendPosition.ENUM.js

```
1       /*
----------------------------------------------------------------------
---------------
2       * Hint: This is a derived (generated) file. Changes should be
done in the underlying
3       * source files only (*.type, *.js) or they will be lost after
the next generation.
4        *
----------------------------------------------------------------------
--------------- */
5
6       // Provides enumeration sap.apb.makit.LegendPosition.
7       jQuery.sap.declare("sap.apb.makit.LegendPosition");
8
9       /**
10       * @class Enumeration for legend position.
11       *
12       * @version 1.0
13       * @static
14       * @public
15       */
16      sap.apb.makit.LegendPosition = {
17
18          /**
```

```
19           * Legend location is on the top of the chart
20           * @public
21           */
22          Top : "Top",
23
24          /**
25           * Legend location is on the left of the chart
26           * @public
27           */
28          Left : "Left",
29
30          /**
31           * Legend location is on the bottom of the chart
32           * @public
33           */
34          Bottom : "Bottom",
35
36          /**
37           * Legend location is on the right of the chart
38           * @public
39           */
40          Right : "Right",
41
42          /**
43           * Hide the legend
44           * @public
45           */
46          None : "None"
47
48        };
49
50
```

makit/SortOrder.ENUM.js

```
/*
---------------------------------------------------------------------------
* Hint: This is a derived (generated) file. Changes should be done in the underlying
* source files only (*.type, *.js) or they will be lost after the next generation.
 *
--------------------------------------------------------------------------- */

// Provides enumeration sap.apb.makit.SortOrder.
jQuery.sap.declare("sap.apb.makit.SortOrder");

/**
 * @class Enumeration for sort order.
 *
 * @version 1.0
 * @static
 * @public
 */
sap.apb.makit.SortOrder = {

    /**
     * Ascending sort
     * @public
     */
    Ascending : "Ascending",

    /**
     * Descending sort
     * @public
```

```
27            */
28           Descending : "Descending",
29
30           /**
31            * No sorting
32            * @public
33            */
34           None : "None"
35
36         };
37
38
```

Index

A

B

C

D

E

F

G

I

L

M

N

P

R

S

T

U