Sitecore Experience Platform 8.1

SPEAK changes from 1.1 to 2.0

Describes differences between SPEAK 1.1 and SPEAK 2.0 in Sitecore 8.1

sitecore®
Own the experience™

Table of contents

Chapter 1 Introduction

Sitecore 8.1 introduces a new version of SPEAK: SPEAK 2.0. Sitecore 8.1 still supports the previous version, SPEAK 1.1, available since Sitecore 7.2. This means that Sitecore can make major improvements to the SPEAK UI framework (in version 2.0) while applications developed for the previous version of SPEAK still work without any changes.

SPEAK 1.1 and SPEAK 2.0 are similar, but there are many differences at a detailed level. A SPEAK page can only use components from one version, and page code developed for SPEAK 1.1 pages has to be modified to work in a SPEAK 2.0 page.

Not all components that are in SPEAK 1.1 are available in the SPEAK 2.0 version for Sitecore 8.1. This means that:

- We recommend that development teams starting new applications targeted for Sitecore 8.1 or later use SPEAK 2.0, as long as this SPEAK version contains the components they need.
- We recommend that development teams already working on SPEAK 1.1 applications wait migrating their applications until the complete set of SPEAK components becomes available in a later version. At that time, Sitecore will provide tools that help developers migrate applications.

This document describes the changes between SPEAK 1.1 and SPEAK 2.0 in Sitecore 8.1. The intended audience is developers that are already familiar with SPEAK 1.1 and who are about to begin work on a new SPEAK 2.0 application.

SPEAK has two parts:

- The core
- The Business Component Library (BCL)

For SPEAK 2.0, the core has changed extensively. Some of these changes are:

- It has been re-architected for ease of use.
- It is easier to create components with new features such as extending objects, ObservableArray, mix-ins and plug-ins, and more.
- It has improved support for reusing groups of components.

SPEAK 2.0 also performs better, and loads and parses pages faster than SPEAK 1.1.

The document contains the following chapters:

- **Chapter 1 – Introduction**
  Introduces the main differences and explains the background.
- **Chapter 2 – Changes for application developers**
  Describes the changes hat are important to consider when you develop SPEAK applications.
- **Chapter 3 – How to create pages and applications with SPEAK 2.0 in Sitecore 8.1**
  Shows how you structure and design applications and pages in SPEAK 2.0.
- **Chapter 4 – Component changes**
  This chapter lists and explains all changes to the SPEAK component.

Chapter 2 Changes for application developers

The status of SPEAK 2.0 as released in Sitecore 8.1 is that around 70% of the BCL 1.1 components are ported to BCL 2.0 in 8.1. This list summarizes the changes to the component library Sitecore made for 8.1:

- Ported most components so that they have the same functionality as they had in SPEAK 1.1
- Renamed some components and properties
- Improved consistency across components so that SPEAK is easier to use for developers
- Added some SPEAK 2.0 only components: **Form, Grid, ConfirmationDialog,** and **ScrollablePanel**
- Implemented some components over from scratch: **ListControl** and **TabControl**

The following subsections describe these changes in some detail. Component changes has concise lists of the changes, component by component.

2.1 The SPEAK version

The changes to the SPEAK core mean that components developed for SPEAK 1.1 are not compatible with the SPEAK 2.0 core, and vice versa. It is possible, but not recommended, to create SPEAK applications where the individual pages do not use the same SPEAK version.

Each SPEAK page has an associated core version set in the **PageCode** component. The SPEAK version for all components in the page must correspond to this SPEAK core version.

Developers set the SPEAK version for a page in the *SpeakCoreVersion* property of the **PageCode** component of the page. There are two options:

- Speak Core 1-1
- Speak Core 2-0

If you do not specify a value, the SPEAK core version defaults to 1.1.

The components for the two SPEAK versions are in these folders:

```
/sitecore/client/Business Component Library/version 1
/sitecore/client/Business Component Library/version 2
```

Sitecore Rocks filters the components based on the `SpeakCoreVersion` when you use **Add Rendering** to add components to a SPEAK page.

2.2 New components

SPEAK 2.0 introduces a number of new components:

- **Form**

  You use the **Form** control for rapid development of responsive end-user input forms with support for object level binding. This creates the foundation for client-side validation and other features in later SPEAK releases.

- **Grid**

  This component allows developers to place components responsively on a page. It replaces the **ColumnPanel** and the **RowPanel** components.

- **ConfirmationDialog**

  This component makes it easy for developers to provide a dialog with a short message to end-users.

- **ScrollablePanel**

  A container for other components with a fixed height. If the child components fill an area larger than the fixed height, the component provides a vertical scrollbar.

2.3 Re-implemented components

Sitecore re-implemented the **ListControl** and **TabControl** components from scratch in SPEAK 2.0. The goals of these re-implementations were:

- To create a more robust architecture
- To provide better usability for both developers and end-users
- To add responsive design features

Note that in Sitecore 8.1, the new **ListControl** only supports the *DetailList* view mode.

The **TabControl** has some known issues in the version that ships with Sitecore 8.1. Refer to the Sitecore 8.1 release notes for details. This is the list of the major issues:

- The control fails to render tabs appropriately in inconsistent situations. Refreshing the page often resolves the issue.
- It may throw a process binding error in some very specific configurations.
- Renderings in lazy loaded tabs (`IsLazyLoaded=true`) must have `PlaceholderKey` set to `Page.Body`. Renderings in non-lazy loaded tabs (`IsLazyLoaded=false`) must have `PlaceholderKey` left blank.
- It ignores the `IsHidden` property on individual tabs.

2.4 Component configuration improvements

In SPEAK 1.1, you often store component configuration details (such as localizable text) in items. Components have a configuration item property, and you specify the GUID of the item that stores the configuration details in this property.

This is tedious to set up. It also makes it difficult to create truly reusable page branches because of the hard coding of the configuration item.

In SPEAK 2.0, SPEAK can find the configuration item using conventions:

- If the configuration item property is blank, SPEAK looks under PageSettings for a configuration item that has the same name as the ID of the component it provides configuration.
- If the configuration item property specifies a GUID, SPEAK uses the item that this GUID points to

The configuration item property also supports absolute and relative item paths.

2.5 StaticData and DynamicData

In SPEAK 2.0 you can specify data for components (for example, the **ListControl**) that show data in two different ways:

- You can point the *StaticData* property to an item in the Sitecore content tree, and the component uses this item as the parent item for the items it shows.
- You can bind the *DynamicData* property to a data source component.

Just as in SPEAK 1.1, you can use a set of tokens to specify the data that is shown. You have the same tokens as in SPEAK 1.1, but two of these tokens have changed names in SPEAK 2.0:

- itemId is now $itemId
- itemName is now $itemName

The full list of tokens you can use in SPEAK 2.0 is:

$database, $displayName, $hasChildren, $icon, $itemId, $itemName, $itemUri, $language, $mediaurl, $path, $templateId, $templateName, $url, and $version.

You can also specify literal names of item fields.

StaticData and standard fields
Note that you cannot use the standard fields when you use *StaticData*. The standard fields are the fields such as `__Created by` or `__Updated`. The names of these fields always begin with __ (two underscores). Note that some of these fields are available as tokens (for example, you can use `$displayName` to get the `__Display name` field).

2.6 Changes and improvements for writing code

The changes to the SPEAK core become most visible to application developers when they write code (in page code, mainly). The following subsections describe the most important.

2.6.1 Assigning values to properties

In SPEAK 1.1 you use get and set methods to retrieve and assign property values:

```
// Assign property value via a set method
app.ComponentID.set("PropertyName", value);
// Retrieve property value via a get method
var value = app.ComponentID.get("PropertyName");
```

In SPEAK 2.0, you assign and retrieve property values directly:

```
// Assign property value via direct assignment
app.ComponentID.PropertyName = value;
// Retrieve property value directly...
var value = app.ComponentID.PropertyName;
// ...or via a property array
var value = app.ComponentID["PropertyName"];
```

Note that the *get* and *set* methods will still work.

2.6.2 Page code changes

In SPEAK 1.1, SPEAK provided an *initialized* method:

```
define(["sitecore"], function (_sc) {
var App =  sc.Definitions.App.extend({
  initialized: function () {}
});
return App;
});
```

In SPEAK 2.0, SPEAK provides a set of lifecycle methods that give developers much more control:

```
 (function(Speak) {
  Speak.pageCode({
   initialize: function() {},
   initialized: function() {},
   beforeRender: function() {},
   render:function() {},
   afterRender:function() {}
});
})(Sitecore.Speak);
```

2.6.3 Component access

In SPEAK 1.1, developers could only access components from **PageCode**.

In SPEAK 2.0, developers can access components in several ways:

- From **PageCode**, as before (`Sitecore.Speak.app`)
- From a parent component (`Sitecore.Speak.app.NestedApp`)
- Or from the global object (`Sitecore.Speak.app.Component`)
- They can also access a deeply nested component with `Sitecore.Speak.app.findApplication()`.

A component has access to the app it belongs to with `this.app`.

2.6.4 Sitecore.Speak.utils

In SPEAK 1.1, SPEAK provided a limited number of utility functions:

```
_.isFunction(object)
$.isArray(object)
```

In SPEAK 2.0, there is a utility functions namespace (`Sitecore.Speak.utils`) with more functions.

Type-checking utilities

You use the type-checking utilities to check if an object is or is not of a specified type. They have a very natural syntax:

`Sitecore.Speak.utils.is.<prefix>.<functionName>(object)`

The <prefix> can be one of these:

- a
- an
- not.a
- not.an

And the `<functionName>` can be one of the following:

- arguments
- array
- boolean
- date
- function
- null
- number
- object
- regexp
- string
- undefined
- empty
- nullorundefined
- guid

This means that you can write expressions like this:

`Sitecore.Speak.utils.is.not.a.number(object)`

and this expression will be `true` when `object` is not a number.

Array utilities

You use the array utilities to manipulate arrays. They have this syntax:

`Sitecore.Speak.utils.array.<functionName>()`

where the <functionName> is one of these:

- toArray(arrayLike) - transforms *arrayLike* (an array-like (DOM) object) to a real Array
- contains(array, obj)– true if *array* contains *object*
- flatten(all, shallow) – flattens the nested array *all*. If *shallow* is `true`, the array is only flattened one level.
- find(obj, predicate, context) – looks through each value in the list and returns the first one that passes a truth test (*predicate*), or undefined if no value passes the test. The function returns as soon as it finds an acceptable element, and doesn't traverse the entire list.

Date utilities

You use the date utilities to convert and manipulate date objects. They have this syntax:

```
Sitecore.Speak.utils.date.<functionName>()
```

where the <functionName> is one of the following:

- toISO(date) – return *date* as an ISO date format string
- parseISO(dateString) – converts *dateString* to an ISO date.
- isISO(date) – return true if *date* is an ISO date format string
- toStringWithFormat(value, format) – *format* specifies the format of the resulting date string (for example: `mm.dd.yyyy`).

Security utilities

You use the security utilities for various functions related to security. There is one security utility:

- Sitecore.Speak.utils.security.antiForgery.getAntiForgeryToken() – This method allows you to get an antiforgery token from the server.

URL utilities

You use the URL utilities for various functions related to URLs. There is one URL utility:

- Sitecore.Speak.utils.url.parameterByName(name) Returns the value of *name* parameter from the URL.

XHR utilities

You use the XHR utilities to create a XMLHttpRequest. There is one XHR utility:

- Sitecore.Speak.utils.xhr – gives you and API for working with AJAX.

2.7 Binding syntax improvements

SPEAK 2.0 extends the binding syntax in a number of ways. It is now possible to use different binding modes (one way, two way, and more), and it is possible to bind to both components and to objects.

2.7.1 Binding modes

SPEAK 2.0 supports the following binding modes:

- One way

  SPEAK initializes the target with the source value and updates it whenever the source is updated. This is how binding works in SPEAK 1.1.

- Two way

  SPEAK initializes the target with the source value, and updates both source and target whenever the other is updated (they are synchronized).

- Two way (non-observable source)

  SPEAK updates the target with the source value, and updates the source whenever the target is updated.

- One time

  SPEAK updates the target with the source value.

- One way to source

  SPEAK does not initialize the target, but it updates the source whenever the target is updated.

2.7.2 Binding to a component

You can bind a property of one component to a property of another component:

- One way: `{Binding Text1.Text}`
- Two way: `{Binding Text1.Text,Mode=TwoWay}`

  There is no special syntax for the "Two way (non-observable source)" mode.

- One time: `{Binding Text1.Text,Mode=OneTime}`
- One way to source: `{Binding Text1.Text,Mode=OneTimeToSource}`

2.7.3 Binding to an observable object

In SPEAK 2.0, you can create bindings programmatically:

- One way: `{Binding $app.person.name}`
- Two way and Two way (non-observable source):
  `{Binding $app.person.name,Mode=TwoWay}`
- One time: `{Binding $app.person.name,Mode=OneTime}`
- One way to source: `{Binding $app.person.name,Mode=OneTimeToSource}`

2.8 Debugger changes

It is no longer necessary to add `sc_debug=1` to the URL to access SPEAK 2.0 components in the browser console.

If you add `sc_debug=1` to the URL, SPEAK 2.0 continues to show the additional information provided in SPEAK 1.1.

Chapter 3 How to create pages and applications with SPEAK 2.0 in Sitecore 8.1

This section describes how you use SPEAK 2.0 as it is available in Sitecore 8.1 to create pages and applications. The emphasis is on the architecture and organization, not the details.

3.1 What is a SPEAK page and a SPEAK application?

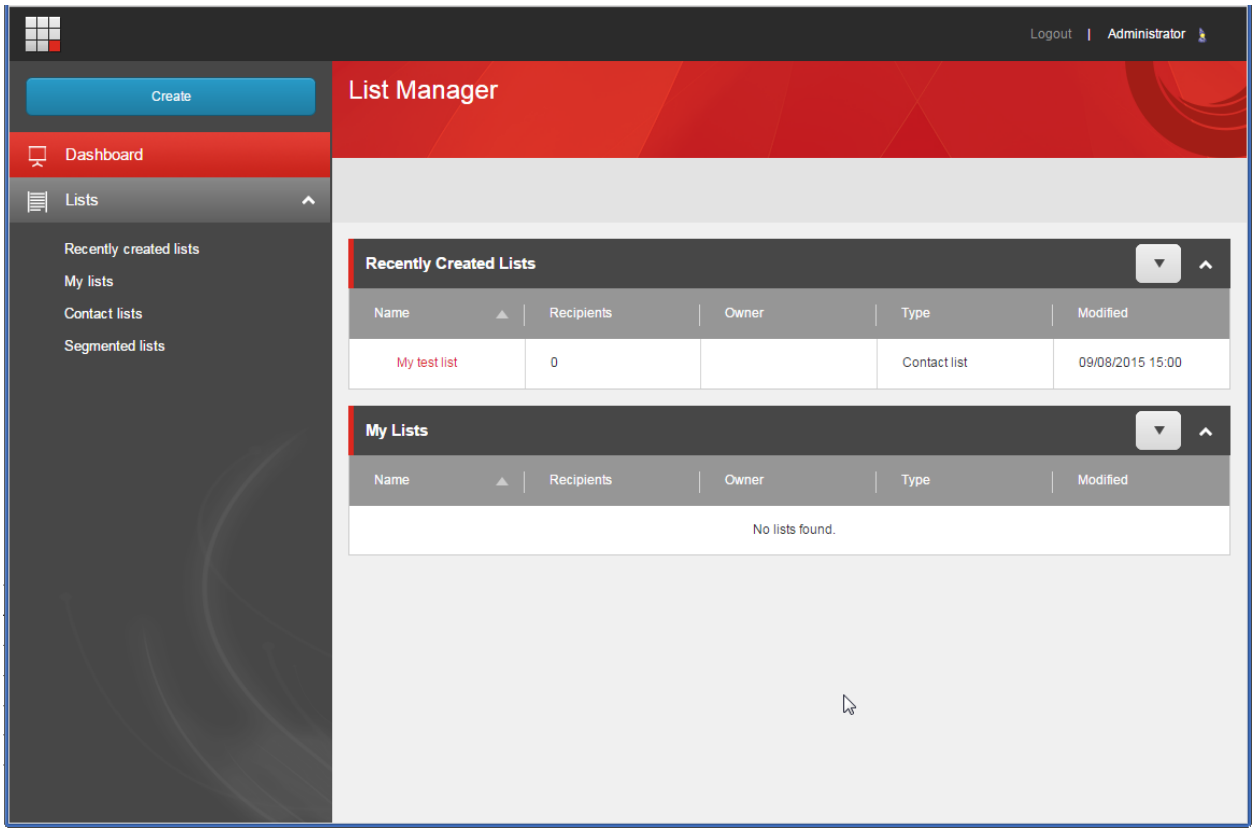
A SPEAK page is a definition item and a layout with SPEAK components. It may or may not include logic, either as front-end JavaScript or as back-end C#.

A SPEAK application is a set of SPEAK pages that enable users to perform a set of related tasks.

The following structure is a summary of UX guidelines for Sitecore applications build in SPEAK. This is not a structure that SPEAK itself imposes.

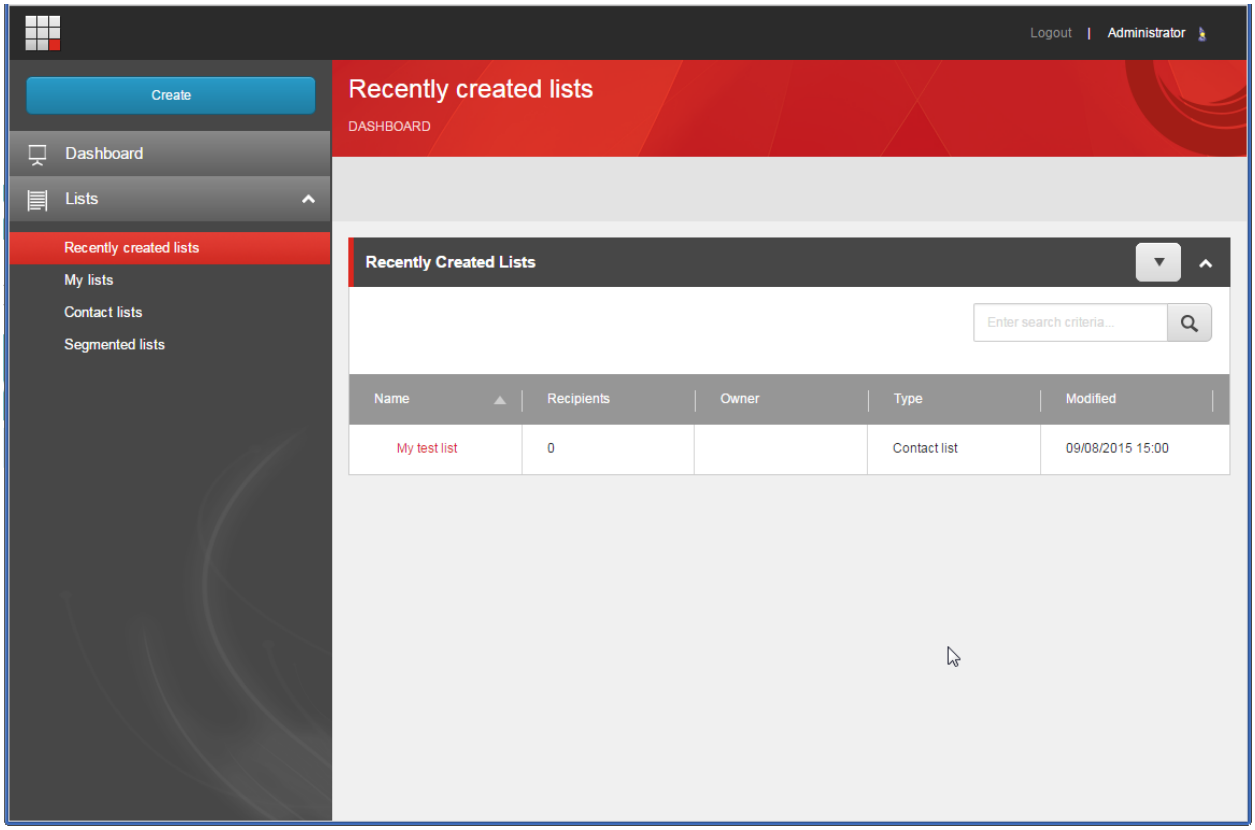
When following these guidelines, a SPEAK application typically has this structure:

- The entry point is a *dashboard*. You typically use the dashboard to give users an overview. You use charts and short lists ("top-5") to give this overview. In some contexts you call what you show in a dashboard KPIs. It is common to have links to full information accompanying the shorter lists. This is an example:

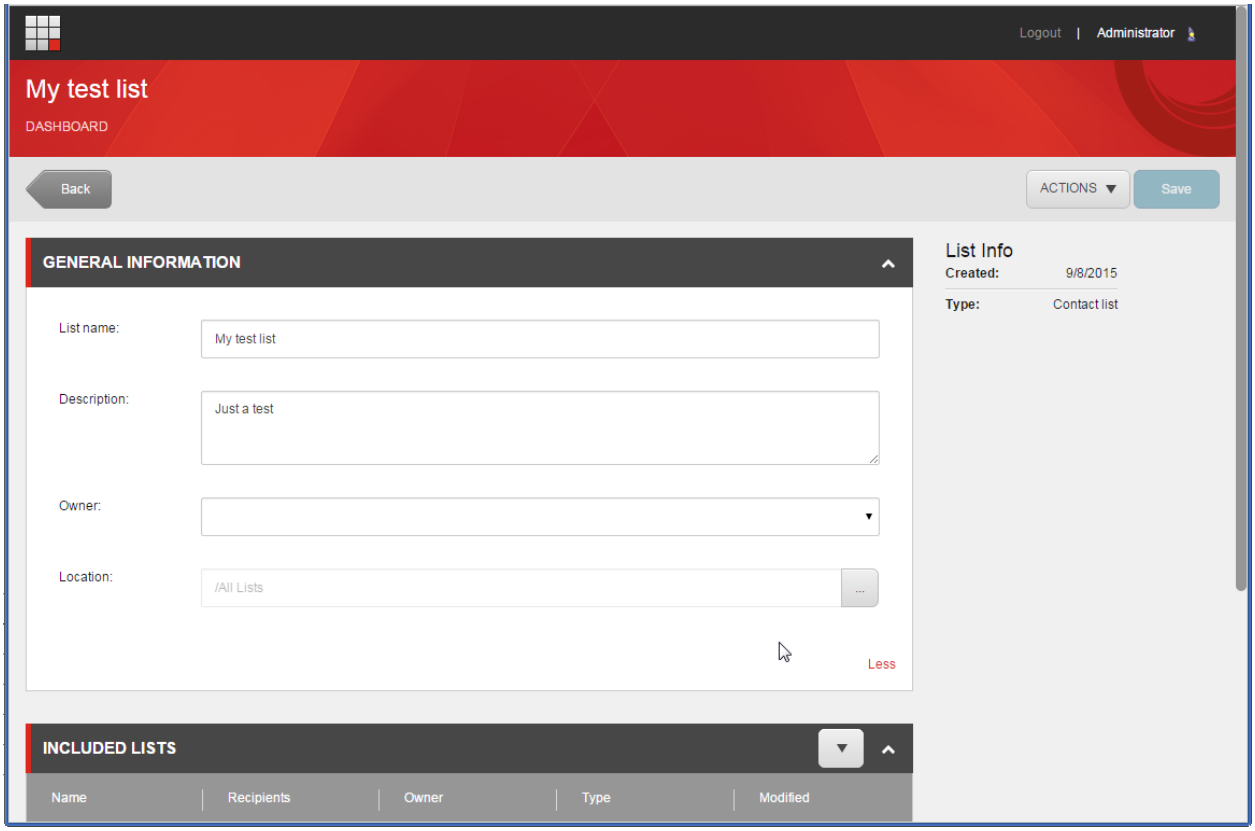


- *List pages* primarily use the **ListControl** to show lists that users can filter and search. It is also a common pattern that users can select in a list and do something with the selection. If, for example,

users see a list of customers, they can select one customer and edit the contact information in a task page. This is an example:

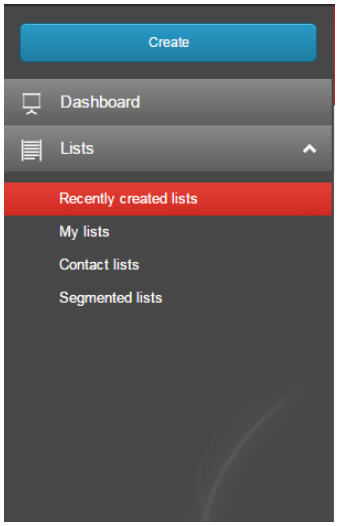


- You create *task pages* for users to work with one item. If, for example, the item is a customer, a task page could be a form where users edit contact information. Users open this task page by clicking a list in the previous example:

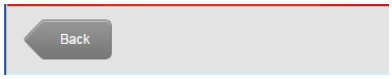


The pages share a global "header" component:

The dashboard and the list page have a navigation menu:



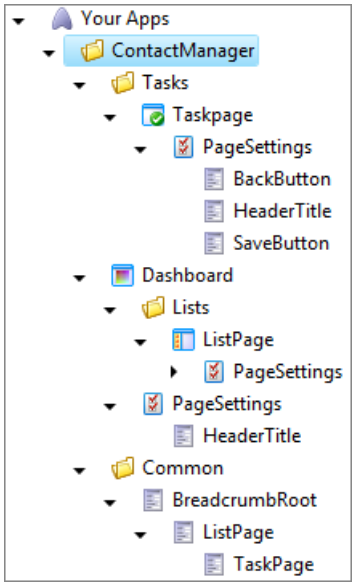
The task page does not have the menu. Instead, there is a **Back** button:



3.2 Structure of dashboard, list and task pages

In many applications you want to show links to the list pages in the menu, but not to the task pages. We recommend that the organization of the items in the content tree reflect this. If you do this, it will be much easier to configure the menu, and it will be much easier to explore the application later.

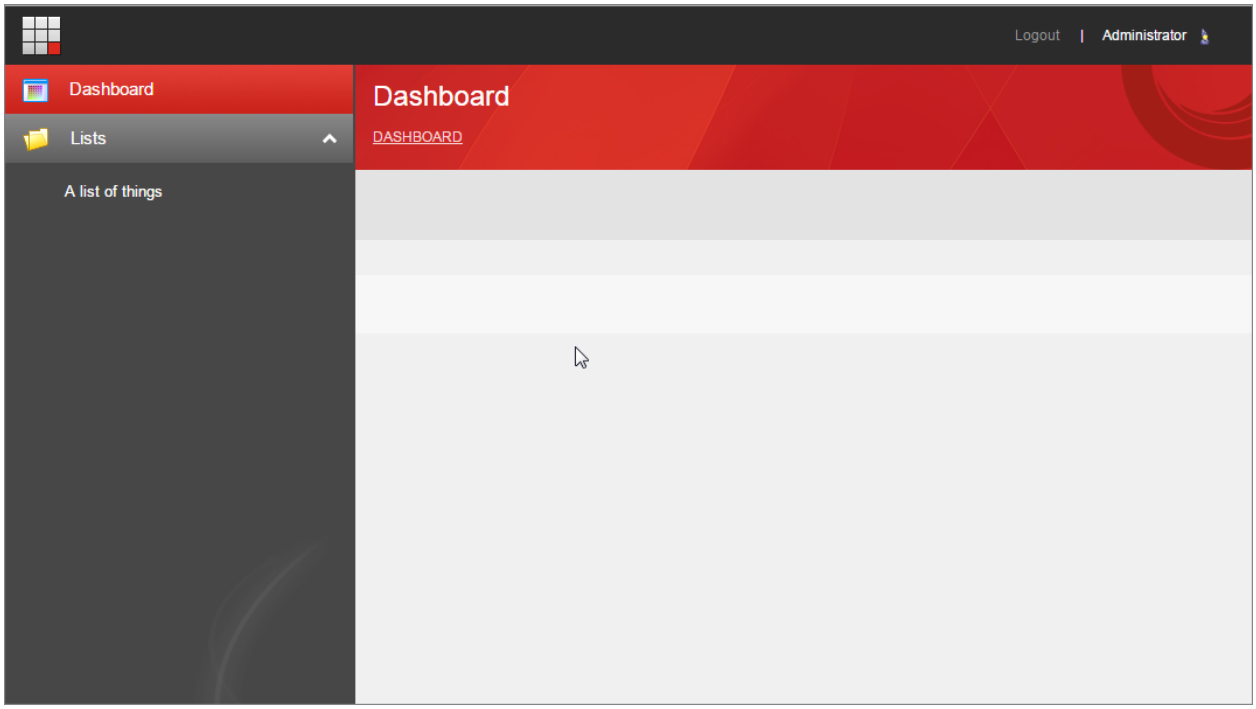
This shows the organization of a very simple application:

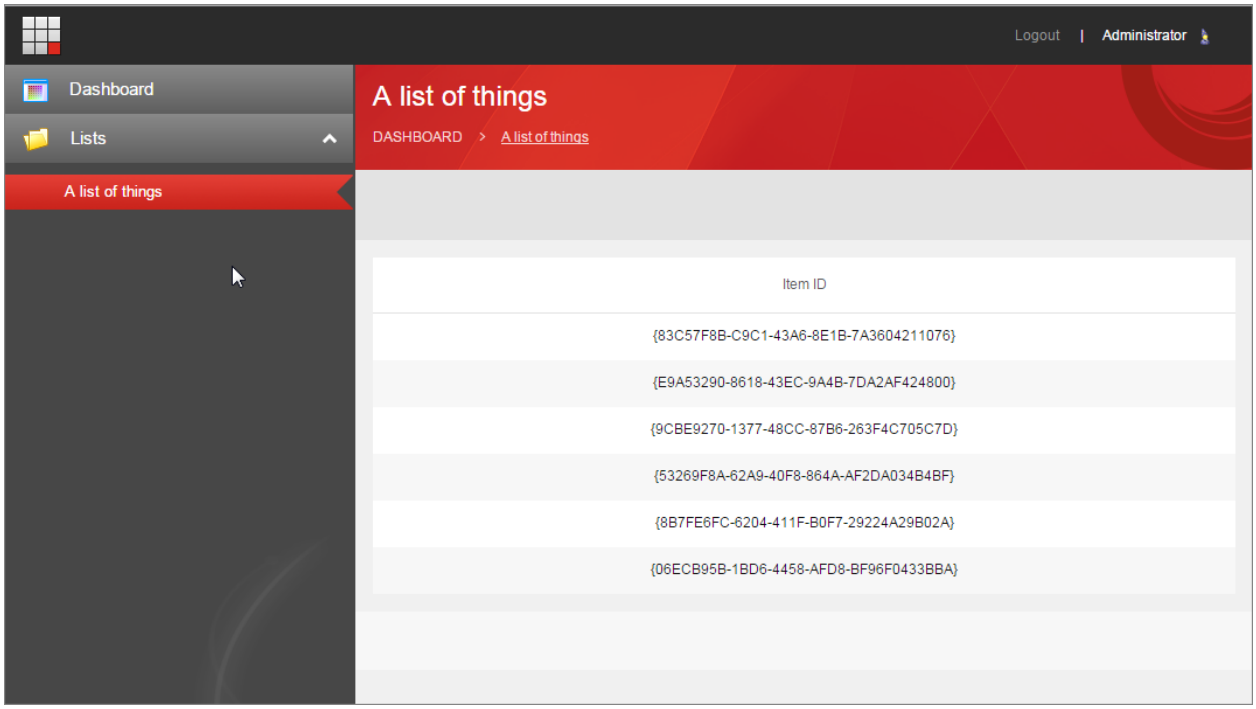


Note the following:

- The application is in a folder ("ContactManager" in this example).
- The list pages are in a **Lists** folder under the **Dashboard** item.
- The task pages are in a **Tasks** folder that is parallel with the **Dashboard** item.
- Each definition item has a **PageSettings** child item.
- The **Menu** control and the **Breadcrumb** control will both show the structure of your application if you point the controls to the **Dashboard** item. If you do not want to have links to task pages in the menu but you do want the breadcrumbs to show task pages as subpages of list pages, you have to create the breadcrumb path manually (this is described in the SPEAK Components Guidance page for the **Breadcrumb** component).

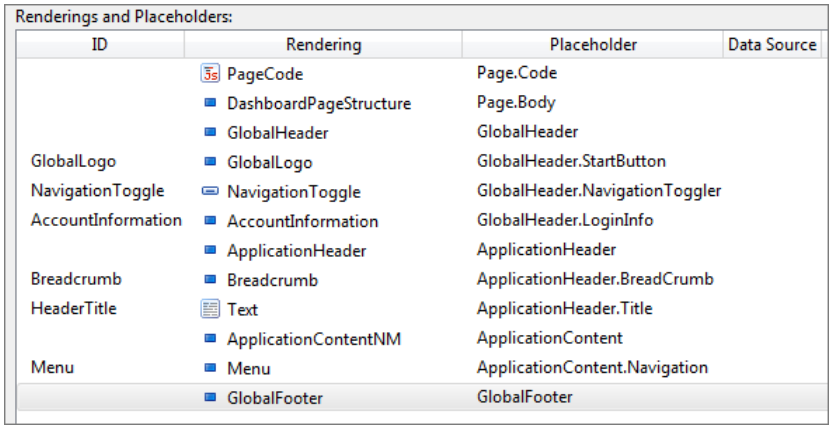
The pages look like this:


Logout | Administrator
Dashboard
Lists
A list of things
Dashboard
DASHBOARD


Logout | Administrator
Dashboard
Lists
A list of things
A list of things
DASHBOARD > A list of things
Item ID
{83C57F8B-C9C1-43A6-8E1B-7A3604211076}
{E9A53290-8618-43EC-9A4B-7DA2AF424800}
{9CBE9270-1377-48CC-87B6-263F4C705C7D}
{53269F8A-62A9-40F8-864A-AF2DA034B4BF}
{8B7FE6FC-6204-411F-B0F7-29224A29B02A}
{06ECB95B-1BD6-4458-AFD8-BF96F0433BBA}

3.3 Design of dashboard, list and task pages

The **Dashboard** page has this layout:



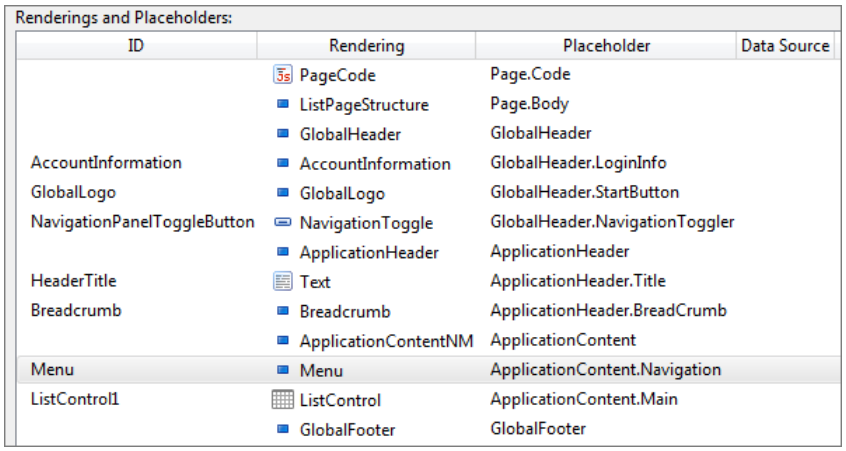
Renderings and Placeholders:

ID	Rendering	Placeholder	Data Source
	PageCode	Page.Code	
	DashboardPageStructure	Page.Body	
	GlobalHeader	GlobalHeader	
GlobalLogo	GlobalLogo	GlobalHeader.StartButton	
NavigationToggle	NavigationToggle	GlobalHeader.NavigationToggler	
AccountInformation	AccountInformation	GlobalHeader.LoginInfo	
	ApplicationHeader	ApplicationHeader	
Breadcrumb	Breadcrumb	ApplicationHeader.BreadCrumb	
HeaderTitle	Text	ApplicationHeader.Title	
	ApplicationContentNM	ApplicationContent	
Menu	Menu	ApplicationContent.Navigation	
	GlobalFooter	GlobalFooter	

The following is noteworthy:

- Set the **SpeakCoreVersion** property of the **PageCode** component to `SpeakCore2-0`.
- The text of the *HeaderTitle* **Text** control comes from the configuration item named **HeaderTitle** under **PageSettings**. When you follow this convention, the **Text** control finds the text without you having to do more.
- The **ApplicationContentNM** page substructure control provides the **ApplicationContent.Navigation** and the **ApplicationContent.Main** placeholders. The navigation placeholder is for the menu, and the main placeholder is for your content.

The **List** page has this very similar layout:



Renderings and Placeholders:

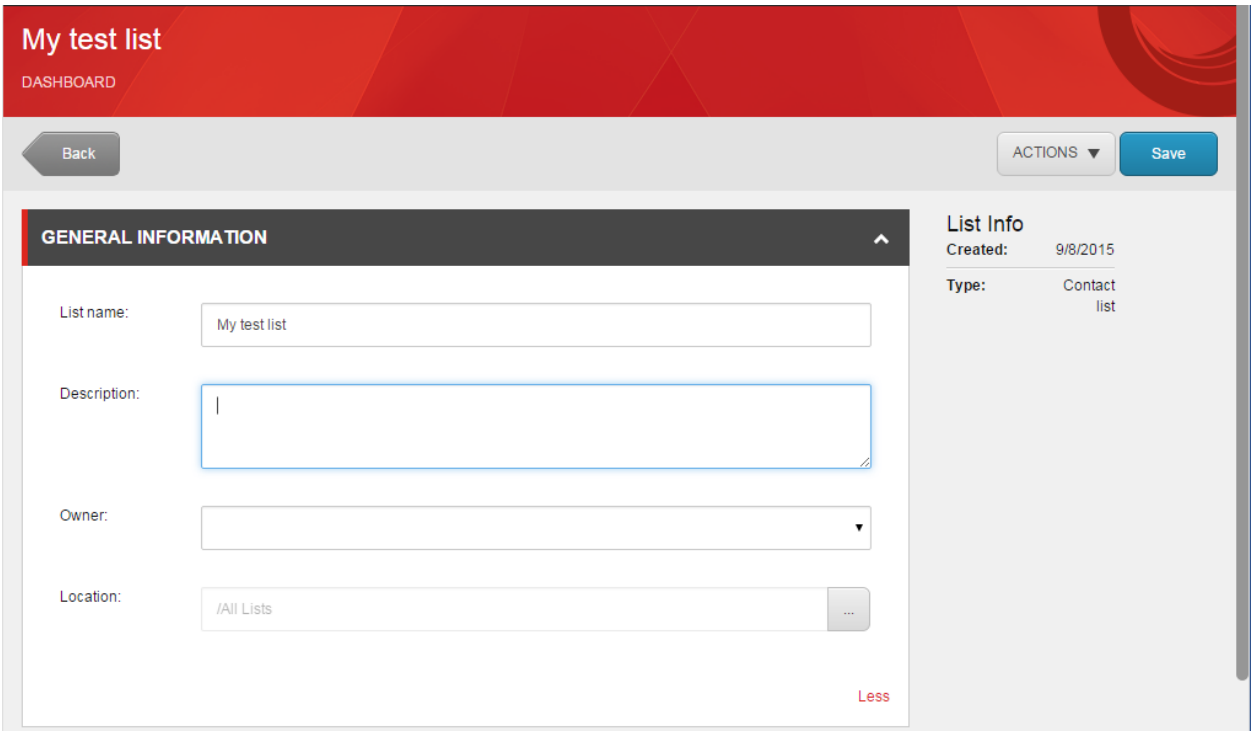
ID	Rendering	Placeholder	Data Source
	PageCode	Page.Code	
	ListPageStructure	Page.Body	
	GlobalHeader	GlobalHeader	
AccountInformation	AccountInformation	GlobalHeader.LoginInfo	
GlobalLogo	GlobalLogo	GlobalHeader.StartButton	
NavigationPanelToggleButton	NavigationToggle	GlobalHeader.NavigationToggler	
	ApplicationHeader	ApplicationHeader	
HeaderTitle	Text	ApplicationHeader.Title	
Breadcrumb	Breadcrumb	ApplicationHeader.BreadCrumb	
	ApplicationContentNM	ApplicationContent	
Menu	Menu	ApplicationContent.Navigation	
ListControl1	ListControl	ApplicationContent.Main	
	GlobalFooter	GlobalFooter	

The **Task** page differs:

Renderings and Placeholders:

ID	Rendering	Placeholder	Data Source
	PageCode	Page.Code	
	TaskPageStructure	Page.Body	
	GlobalHeader	GlobalHeader	
GlobalLogo	GlobalLogo	GlobalHeader.StartButton	
AccountInformation	AccountInformation	GlobalHeader.LoginInfo	
	GlobalFooter	GlobalFooter	
	ApplicationHeader	ApplicationHeader	
HeaderTitle	Text	ApplicationHeader.Title	
Breadcrumb	Breadcrumb	ApplicationHeader.BreadCrumb	
BackButton	BackButton	ApplicationHeader.Back	
SaveButton	Button	ApplicationHeader.Actions	
	ApplicationContentMI	ApplicationContent	

This page uses the **ApplicationContentMI** page substructure, and it does not have a menu. The page substructure provides two placeholders: **ApplicationContent.Main** for you main content, and **ApplicationContent.Info** that sets aside an area to the right of the main content area. You use this area for additional information, as in this example:



Chapter 4 Component changes

This section describes the changes to the SPEAK components. It has three subsections:

- Changes compared to SPEAK 1.1 components. This subsection lists all SPEAK 1.1 components that have SPEAK 2.0 equivalents but where the SPEAK 2.0 version is different in some way.
- Components introduced in SPEAK 2.0.
- SPEAK 1.1 Components not yet ported to SPEAK 2.0.

4.1 Changes compared to SPEAK 1.1 components

Apart from the changes listed in the table below, there is a general change:

- Many components have a new property called **ConfigurationItem**. It points to a configuration item for the component.
- However, a certain number of components have a mistake, and the **StaticData** property points to the configuration item instead. This mistake will be corrected in the next release of SPEAK 2.0. The following components have this mistake:
  - **Breadcrumb**
  - **Form**
  - **MessageBar**
  - **Menu**
  - **TabControl**

Component (1.1 name)	Changes
AdvancedExpander	• Renamed **Expander**.
AreaChart	• The **Items** property renamed **DynamicData**.
ArrowIndicator	• The options for the **ImageSize** property changed.
BackButton	• The **Text** property moved to different section.
BarChart	• The **Items** property renamed **DynamicData**.
Border	• Incorporates the functionality of the deprecated **ProtectedBorder** control. • New properties: **RequiresRead** and **RequiresWrite**. • The **ContentAlignment** property renamed **ContentHorizontalAlignment**.
Breadcrumb	• The **RootItemtId** property renamed **StaticData**.
Button	• New property **IsFullWidth**. • The **Text** property moved to different section. • The **Dimension** property was removed. • The **BackgroundPosition** property renamed .**SpritePosition.**
ButtonTextBox	• The **IsReadonly** property renamed **IsReadOnly**. • New properties: **KeyboardType, InputType** and **SpritePosition**. • The **ButtonImageUrl** property renamed **Icon**. • The **Text** property renamed **Value** and moved to a new section.
ChartDataProvider	• Component renamed **ChartDataSource**.
CheckBox	• New event: **ValueModified**. • The **IsChecked** property has a new field type. • The **Text** property renamed **Label**.
ColumnChart	• The **Items** property renamed **DynamicData**.

Component (1.1 name)	Changes
ComboBox	• Component renamed **DropList**. • The **Items** property renamed **DynamicData**. • New property: **StaticData**. • Added a JavaScript API for selection. • The *addItem* JavaScript method removed.
Dashboard	• Component renamed **DashboardPageStructure**.
Dialog	• Component renamed **DialogPageStructure**.
DialogWindow	• The **Size** property has a new option: *StretchToFill*. • The **MessageSpecific** option for the Size property removed.
DoughnutChart	• The **Items** property renamed **DynamicData**.
Expander	• Deprecated, use the 2.0 **Expander** instead.
Form	• Deprecated, use the 2.0 **FormControl**.
Frame	• The **Height** property moved to a different section. • The **Width** property removed.
GenericDataProvider	• Component renamed **GenericDataSource**. • The **HasData**, **HasNoData**, **IsBusy**, **PageSize**, and **TotalRecordCount** properties moved to other sections. • The **Item** property renamed **DynamicData**. • The **PageNumber** property renamed **PageSize**.
Image	• **MediaItemDataSource** property added.
HyperlinkButton	• New property: **Icon**. • The **Text** property moved to a different section.
JourneyChart	• Deprecated, use **Timeline**.
LineChart	• The **Items** property renamed **DynamicData**.
List	• Component renamed **ListPageStructure**.
ListBox	• New properties: **SelectedItemIds**, **Size**, **StaticData**, and **ValueModified**. • **IsMultipleSelectedEnabled** property moved to another section. • **Selection** property removed. • The **Items** property renamed **DynamicData**. • The **Multiple** property renamed **IsMultipleSelectedEnabled**.
ListControl	• Deprecated, use the 2.0 **ListControl**.
Menu	• New properties: **SelectedItem** and **SelectedItemId**. • The **ItemDataSource** property renamed **StaticData**. • Options for the **ExpandedItemIds** property now use "|" (pipe) as separator.
MultiSelectList	• Deprecated.
NavigationPanelToggleButton	• Component renamed **NavigationToggle**.
PieChart	• The **Items** property renamed **DynamicData**.
Pipeline	• The **TargetControl** property now has bindmode "server". • The **LoadMode** property moved to another section.

Component (1.1 name)	Changes
ProtectedBorder	• Deprecated. Use the **Border** control.
QueryDataSource	• The **HasItems** property renamed **HasData**. • The **HasMoreItems** property renamed **HasMoreData**. • The **HasNoItems** property renamed **HasNoData**. • The **HasData, HasNoData,** and **IsBusy** properties moved to a new section.
RadarChart	• The **Items** property renamed **DynamicData**.
RowPanel	• Deprecated.
Rule	• The **LoadMode** property moved to another section.
Scrollbar	• Deprecated.
SearchDataSource	• New JavaScript method: *loadData*. • New design-time JavaScript property: *isLoadDataDeferred*. • The **HasItems** property renamed **HasData**. • The **HasMoreItems** property renamed **HasMoreData**. • The **HasNoItems** property renamed **HasNoData**. • The **HasData, HasNoData,** and **IsBusy** properties moved to other sections. • The **Text** property renamed **SearchQuery**. • The **RootItemId** property renamed **SearchRootItemId**.
SearchResults	• Deprecated.
TabControl	• Deprecated, use the 2.0 **TabControl**.
Task	• Component renamed **TaskPageStructure**.

4.2 Components introduced in SPEAK 2.0

The following components are new in SPEAK 2.0:

- ConfirmationDialog
- Grid
- ScrollablePanel
- SearchableDropList

4.3 SPEAK 1.1 Components not yet ported to SPEAK 2.0

These components do not have a SPEAK 2.0 version in Sitecore 8.1:

- ActionControl
- AdvancedComboBox
- Carousel
- ColumnPanel
- ContextSwitcher
- DatePicker
- DropDownButton
- FilterControl
- HyperlinkButtonsGroup
- IconButton
- IconHyperlinkButton
- ItemTreeView
- JourneyChart
- Label
- LoadOnDemandPanel
- Popover
- ProgressBar
- ProgressIndicator
- ProtectedBorder
- RadioButton
- Repeater
- RowPanel
- SearchPanel
- Section
- SettingsDictionary
- SilverlightFrame
- Slider
- SmartPanel
- StringDictionary
- Timeline
- TimePicker
- ToolTip
- UploaderInfo
- WebServiceDataSource