Sitecore CMS 7.2

SPEAK Developer's Cookbook

Tips and Techniques for Development Teams Creating SPEAK Applications

Table of Contents

Chapter 1

Introduction

This cookbook provides practical advice, tips, and techniques to help development teams creating business applications in SPEAK. The cookbook is not complete, and it will be replaced by a Developer's Guide for a later version of Sitecore.

This document contains the following chapters:

- Chapter 1 – **Introduction**
  This chapter; an introduction to this document and to SPEAK.
- Chapter 2 – **Recipes**
  Instructions for performing common activities.
- Chapter 3 – **Q&A**
  Answers to frequently asked questions.
- Chapter 4 – **Order Management Application**
  Description of a simple self-study application.

1.1 About SPEAK

SPEAK (*Sitecore Process Enablement & Accelerator Kit*) is a development framework based on Sitecore. It lets you develop Sitecore applications with a consistent interface for marketing and business users quickly and easily.

SPEAK gives you a predefined set of page layouts and components and also a predefined set of user experience guidelines.

SPEAK offers an alternative to SHEER UI and .NET aspx pages, which have been popular approaches to creating applications in Sitecore.

Sitecore will use SPEAK to create future applications and as the framework for future enhancements to the current suite of applications.

1.1.1 Required Skills

Developers using SPEAK will need basic Sitecore development skills, including knowledge of the following concepts and tools:

- Items
- Templates
- Layouts
- Renderings
- Placeholders
- Rules
- Visual Studio
- Sitecore Rocks
- C# (only for implementing advanced business logic)
- JavaScript (only for implementing advanced business logic)

1.1.2 SPEAK History

The original motivation for SPEAK came after Sitecore released the digital marketing platform and began introducing more applications for business and marketing users. The traditional Sitecore applications were designed for developers and content authors. Feedback from marketing and business users suggested an opportunity for a streamlined and consistent user interface that offered more guidance, organized related activities, and that took advantage of newer browser-based technologies.

Sitecore has invested several years in creating a new user experience paradigm specifically designed for the needs of business and marketing users.

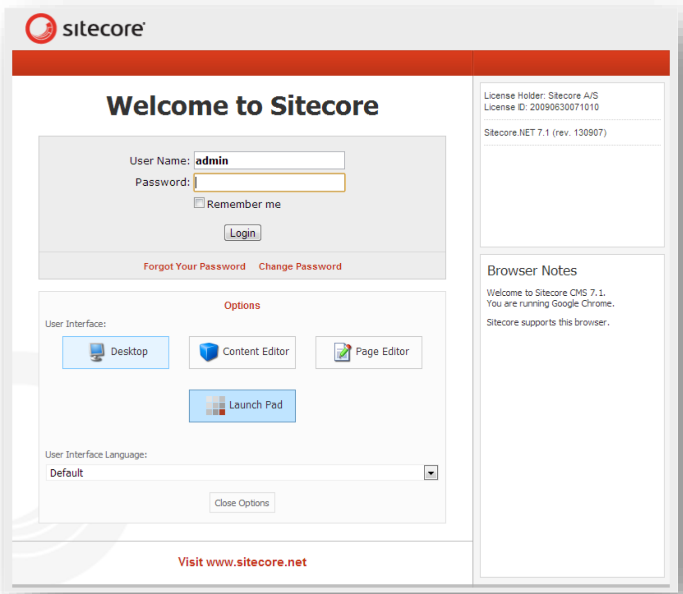
The SPEAK UI framework included in Sitecore CMS 7.2 represents the third generation of the framework.

Sitecore is currently working on extending SPEAK for future releases of Sitecore CMS.

1.2 SPEAK Applications

End users access SPEAK applications via the SPEAK Launch Pad, available as an option in the standard Sitecore login page as of CMS 7.1.

The screen shot below shows the Launch Pad option highlighted.



Welcome to Sitecore

User Name: admin

Password:

Remember me

Login

Forgot Your Password Change Password

Options

User Interface:

Desktop Content Editor Page Editor

Launch Pad

User Interface Language:

Default

Close Options

Visit www.sitecore.net

License Holder: Sitecore A/S
License ID: 20090630071010

Sitecore.NET 7.1 (rev. 130907)

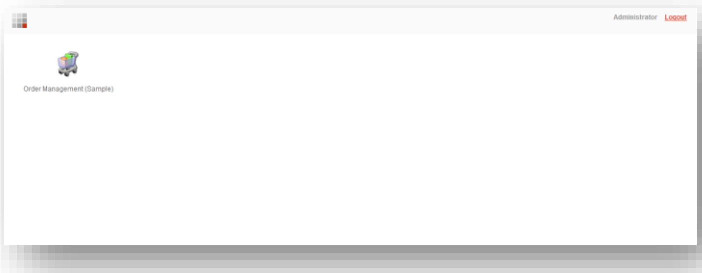
Browser Notes

Welcome to Sitecore CMS 7.1.
You are running Google Chrome.

Sitecore supports this browser.

The SPEAK Launch Pad displays application short cuts for any installed SPEAK applications. By default, CMS 7.2 does not provide any SPEAK applications, so the Launch Pad does not display any shortcuts until an administrator installs one or more SPEAK applications.

The screen shot below shows the Launch Pad with a single application installed.



1.3 Application Page Types

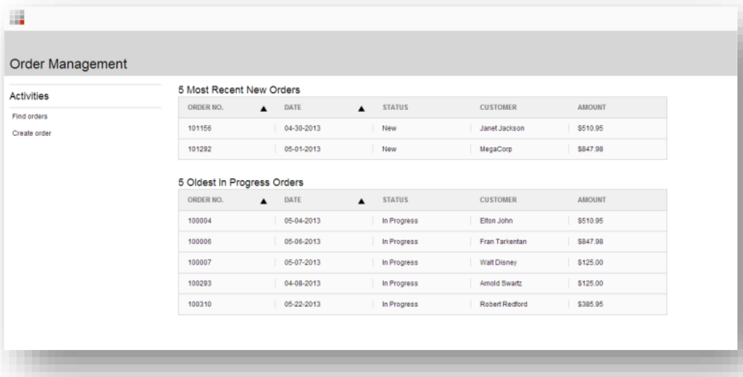
SPEAK supports three main page types:

- Dashboard pages
- List pages
- Task pages

1.3.1 Dashboard Pages

Many applications display a Dashboard page as the start page. Dashboard pages provide an overview of the information managed by the application, as well as navigation options to access information that is more specific.

The screen shot below shows the example Order Management application’s start page, which is itself a very simple Dashboard page.



Order Management

5 Most Recent New Orders

ORDER NO. ▲	DATE ▲	STATUS	CUSTOMER	AMOUNT
101156	04-30-2013	New	Janet Jackson	$510.95
101292	05-01-2013	New	MegaCorp	$847.98

5 Oldest In Progress Orders

ORDER NO. ▲	DATE ▲	STATUS	CUSTOMER	AMOUNT
100004	05-04-2013	In Progress	Elton John	$510.95
100006	05-06-2013	In Progress	Fran Tarkentan	$847.98
100007	05-07-2013	In Progress	Walt Disney	$125.00
100293	04-08-2013	In Progress	Arnold Swartz	$125.00
100310	05-22-2013	In Progress	Robert Redford	$385.95

Dashboards will typically provide charts and other summary information that help the marketing users assess the effectiveness of their work and locating entities that require attention.

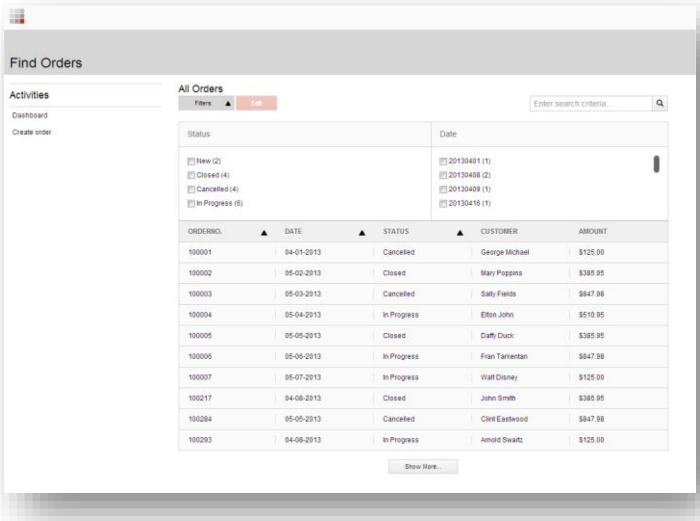
Applications may offer multiple dashboards, both as an application start page and for additional, more specific areas of the application.

Dashboard pages, by convention, do not provide controls that allow users to modify or manipulate entities. Instead, the dashboard page will allow the user to navigate to a Task page that provides editing capabilities for a selected entity. See the Task Pages section below for more information about task pages.

1.3.2 List Pages

List pages provide a mechanism for locating entities that the application manages. List pages typically provide both search and filtering controls to assist the user in finding the desired entities. Unlike dashboard pages, which may display multiple lists, list pages provide a single list that fills the entire page. Like dashboard pages, by convention, list pages do not provide editing facilities, but rather provide navigation to a task page for a selected entity.

The screen shot below shows the example Order Management application’s Find Orders page, which is a simple List page.



Find Orders

Activities

Dashboard

Create order

All Orders

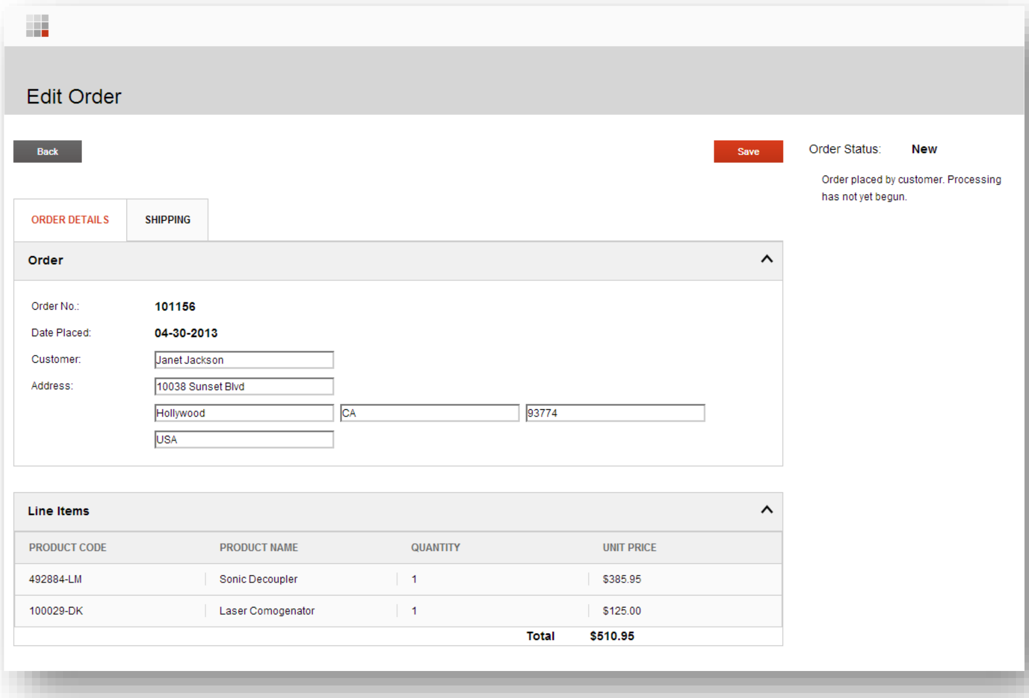
Status	Date
☐ New (2)	☐ 20130401 (1)
☐ Closed (4)	☐ 20130408 (2)
☐ Cancelled (4)	☐ 20130409 (1)
☐ In Progress (6)	☐ 20130416 (1)

ORDERNO. ▲	DATE ▲	STATUS ▲	CUSTOMER	AMOUNT
100001	04-01-2013	Cancelled	George Michael	$125.00
100002	05-02-2013	Closed	Mary Poppins	$385.95
100003	05-03-2013	Cancelled	Sally Fields	$847.98
100004	05-04-2013	In Progress	Elton John	$510.95
100005	05-05-2013	Closed	Daffy Duck	$385.95
100006	05-06-2013	In Progress	Fran Tarkentan	$847.98
100007	05-07-2013	In Progress	Walt Disney	$125.00
100217	04-08-2013	Closed	John Smith	$385.95
100284	05-05-2013	Cancelled	Clint Eastwood	$847.98
100293	04-08-2013	In Progress	Arnold Swartz	$125.00

1.3.3 Task Pages

Task pages provide tools that allow business users to manipulate entities. Task pages do not provide navigation to other pages (other than back to the previously seen dashboard or list page).

The screen shot below shows the example Order Management application's Edit Order page.



Edit Order

Order Status: **New**

Order placed by customer. Processing has not yet begun.

Order

Order No.:	**101156**		
Date Placed:	**04-30-2013**		
Customer:	Janet Jackson		
Address:	10038 Sunset Blvd		
	Hollywood	CA	93774
	USA		

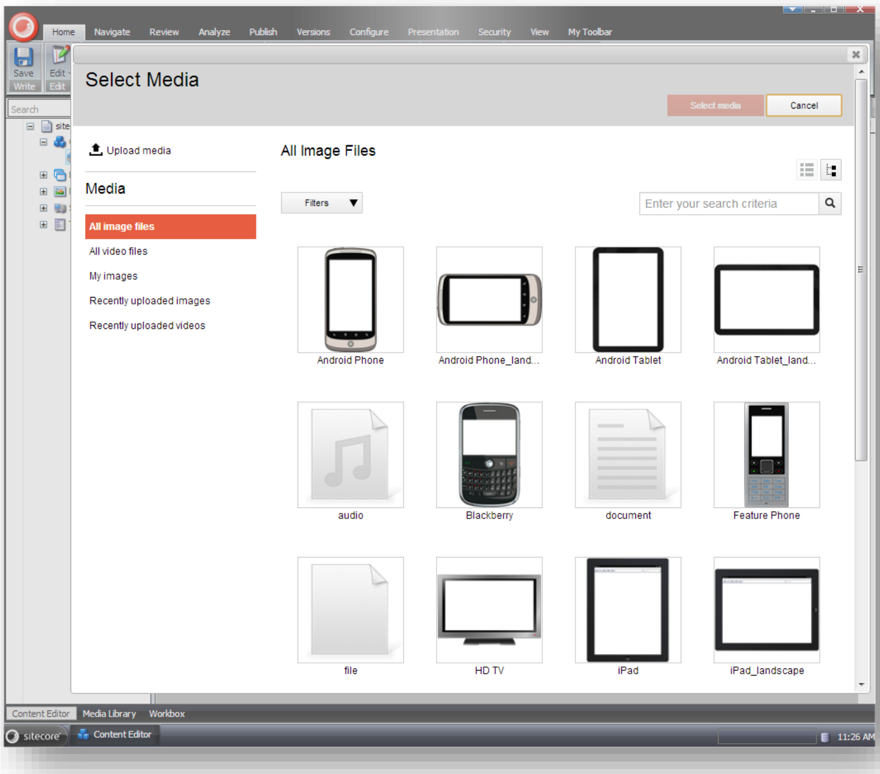
Line Items

PRODUCT CODE	PRODUCT NAME	QUANTITY	UNIT PRICE
492884-LM	Sonic Decoupler	1	$385.95
100029-DK	Laser Comogenator	1	$125.00
		Total	**$510.95**

1.4 SPEAK Dialogs

Developers may also use SPEAK to develop dialogs for use in the traditional Sitecore client or in SPEAK applications.

The screen shot below shows the Select Media dialog which the Content Editor displays when the user clicks Browse for an image field.



Chapter 2

Recipes

This chapter provides instructions describing how to perform frequently performed activities.

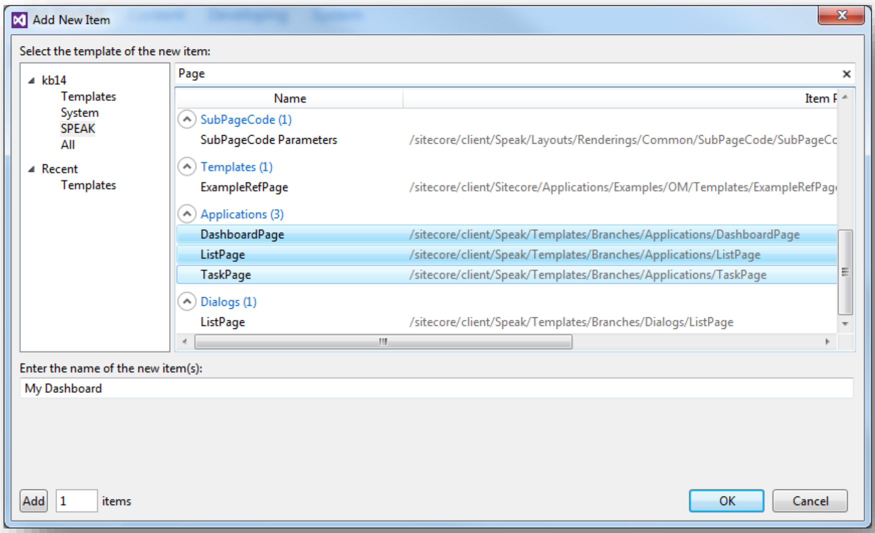
2.1 Creating a Page

Before actually creating a page, you should decide a number of things about the page, including:

- Where will you store the page's "definition item"?
- What type of page you will create?
- What name will you give your page definition item?
- Remember: you cannot copy an existing page as a starting point.

When you know the answers to the above questions, open Sitecore Rocks.

1. Locate and select the desired parent for the new page definition item.
2. Right click on the item and choose Add > New Item… from the menu (or click Ctrl+A).
3. Make sure you have SPEAK selected in the left pane.
4. Select the branch associated with the page type you want to create (the three page type branches are shown highlighted in the image below).
5. Type the name for your page.
6. Click OK.



Select the template of the new item:

Page

Name	Item P
SubPageCode (1)	
SubPageCode Parameters	/sitecore/client/Speak/Layouts/Renderings/Common/SubPageCode/SubPageCo
Templates (1)	
ExampleRefPage	/sitecore/client/Sitecore/Applications/Examples/OM/Templates/ExampleRefPag
Applications (3)	
DashboardPage	/sitecore/client/Speak/Templates/Branches/Applications/DashboardPage
ListPage	/sitecore/client/Speak/Templates/Branches/Applications/ListPage
TaskPage	/sitecore/client/Speak/Templates/Branches/Applications/TaskPage
Dialogs (1)	
ListPage	/sitecore/client/Speak/Templates/Branches/Dialogs/ListPage

Enter the name of the new item(s):

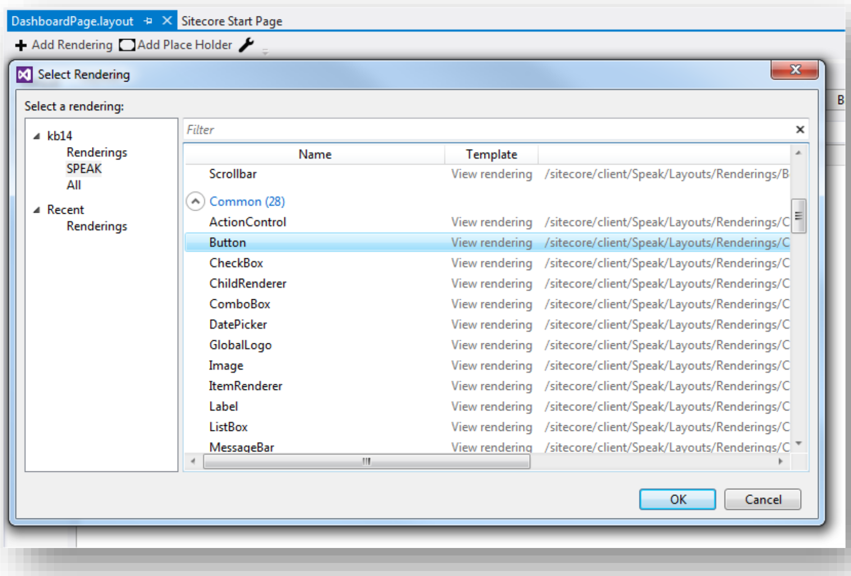
My Dashboard

1

2.2 Adding Components to a Page

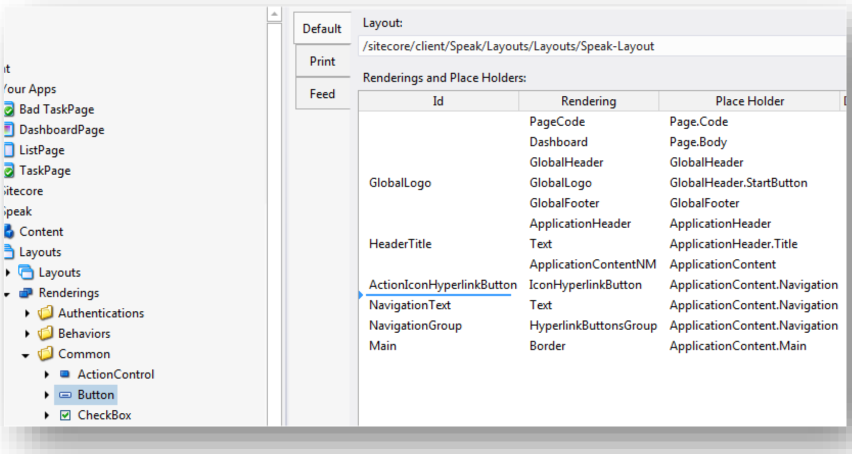
There are two ways to add components to the page:

- Use the Add Rendering button in the Rocks Layout Designer.
  - This provides a search window and adds the component to the end of the list of components.



Name	Template	
Scrollbar	View rendering	/sitecore/client/Speak/Layouts/Renderings/B
Common (28)		
ActionControl	View rendering	/sitecore/client/Speak/Layouts/Renderings/C
Button	View rendering	/sitecore/client/Speak/Layouts/Renderings/C
CheckBox	View rendering	/sitecore/client/Speak/Layouts/Renderings/C
ChildRenderer	View rendering	/sitecore/client/Speak/Layouts/Renderings/C
ComboBox	View rendering	/sitecore/client/Speak/Layouts/Renderings/C
DatePicker	View rendering	/sitecore/client/Speak/Layouts/Renderings/C
GlobalLogo	View rendering	/sitecore/client/Speak/Layouts/Renderings/C
Image	View rendering	/sitecore/client/Speak/Layouts/Renderings/C
ItemRenderer	View rendering	/sitecore/client/Speak/Layouts/Renderings/C
Label	View rendering	/sitecore/client/Speak/Layouts/Renderings/C
ListBox	View rendering	/sitecore/client/Speak/Layouts/Renderings/C
MessageBar	View rendering	/sitecore/client/Speak/Layouts/Renderings/C

- Drag and Drop a component from the Content Tree to the Rock Layout Designer.
  - This allows you to specify the position in the list where you want to place the component.



Layout:
/sitecore/client/Speak/Layouts/Layouts/Speak-Layout

Renderings and Place Holders:

Id	Rendering	Place Holder
	PageCode	Page.Code
	Dashboard	Page.Body
	GlobalHeader	GlobalHeader
GlobalLogo	GlobalLogo	GlobalHeader.StartButton
	GlobalFooter	GlobalFooter
	ApplicationHeader	ApplicationHeader
HeaderTitle	Text	ApplicationHeader.Title
	ApplicationContentNM	ApplicationContent
ActionIconHyperlinkButton	IconHyperlinkButton	ApplicationContent.Navigation
NavigationText	Text	ApplicationContent.Navigation
NavigationGroup	HyperlinkButtonsGroup	ApplicationContent.Navigation
Main	Border	ApplicationContent.Main

2.3 Configuring a ListControl

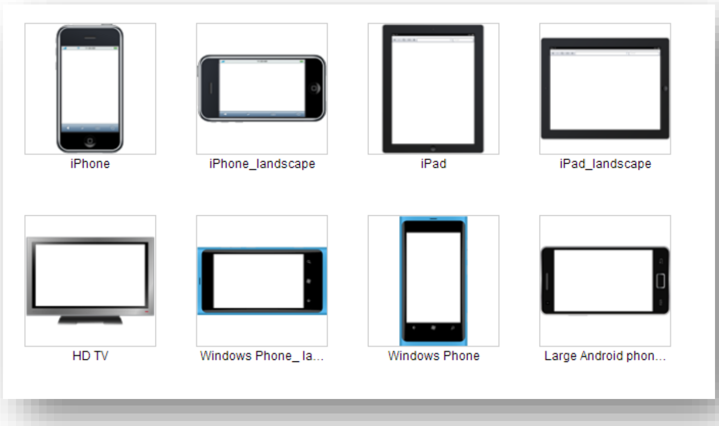
When you add a ListControl to a page, the control initially only shows a gray box. You must perform a number of steps to configure the control to show the information you want. These steps include:

- Define the control's ViewMode as either IconList or DetailList
- Define the columns for the control (if the control will support the DetailList ViewMode).
- Specify where the content of the control will come from.
- [Optional] Make one or more columns sortable.
- [Optional] Format the content of one or more columns.

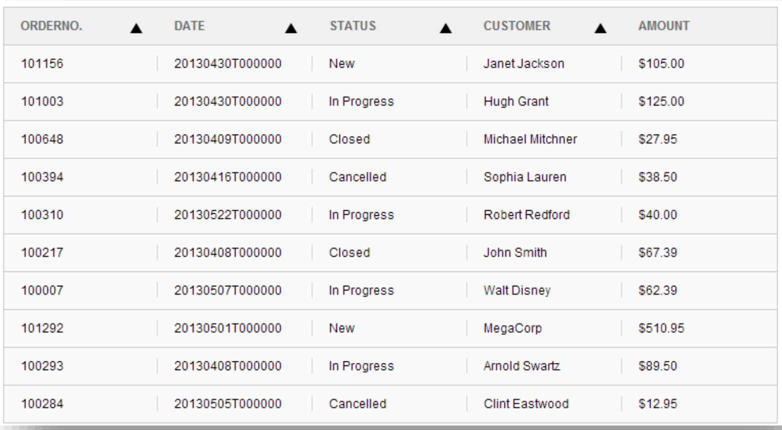
2.3.1 ViewMode

The ListControl supports two view modes: IconList and DetailList.

IconList shows the ListControl content as a grid of images and associated titles, as shown below:



DetailList shows the ListControl content as a series of rows and columns, as shown below:



ORDERNO. ▲	DATE ▲	STATUS ▲	CUSTOMER ▲	AMOUNT
101156	20130430T000000	New	Janet Jackson	$105.00
101003	20130430T000000	In Progress	Hugh Grant	$125.00
100648	20130409T000000	Closed	Michael Mitchner	$27.95
100394	20130416T000000	Cancelled	Sophia Lauren	$38.50
100310	20130522T000000	In Progress	Robert Redford	$40.00
100217	20130408T000000	Closed	John Smith	$67.39
100007	20130507T000000	In Progress	Walt Disney	$62.39
101292	20130501T000000	New	MegaCorp	$510.95
100293	20130408T000000	In Progress	Arnold Swartz	$89.50
100284	20130505T000000	Cancelled	Clint Eastwood	$12.95

If you have data that could appear in either mode, you can allow users to switch between modes.

You specify the default view mode in one of two places:

- On the control definition in the ViewMode property in the Rocks Layout Designer.
- On the configuration item for the control in the DefaultView field.

2.3.2 DetailList Columns

Developers define the columns shown in a ListControl in configuration items. The ListControl displays content based on a set of items. The columns correspond to fields in the items. The ListControl shows the columns when the ViewMode is set to DetailList.

1. Developers create a configuration item based on the ListControl template.
   - Copy the configuration item's GUID and paste it into the ListControl's DataSource property.
2. For each desired column…
   - Developers create a configuration item based on the ColumnField template.
   - Developers define the HeaderText, which appears at the top of the column in the ListControl.
   - Developers specify the field name for the data shown in the column, writing the field name as it appears in the corresponding template for the items shown in the ListControl.

2.3.3 ListControl Content

ListControls display items. Developers configure which items to display in the Items property of the ListControl in the Rocks Layout Designer.

For example, developers often choose to bind the ListControl's Items property to a DataSource control's Items property. SPEAK automatically displays the items found by the bound SearchDataSource component in the ListControl.

For example, in the Example Order Management application, the RecentNewOrdersList on the Start Page sets the Items property to {Binding NewOrdersDataSource.Items}. This means that this ListControl will show the items found by that SearchDataSource component.

The NewOrdersDataSource component defines a RootItemId property, which tells SPEAK where to begin its searches in the content tree, and fills the SearchConfigItemID property with the GUID of a Search Config item, which indicates which items to retrieve by providing various default filters. In the example app, the New Orders Search Config item indicates that the SearchDataSource should only return items based on the Order template that appear under the /sitecore/Client/Sitecore/Applications/Examples/OM/Fake Data/Orders item.

2.3.4 Sortable Columns

To enable sorting, you must make one or more columns sortable and set up sorting in the DataSource that provides items to the ListControl.

The ColumnField template, used to create the ListControl column configuration item, includes a Sortable toggle. Click the field to place a tick mark in the toggle to enable sorting for any given column.

In the SearchDataSource that provides items, set the Sorting property to bind to the ListControl's Sorting property. This allows the sorting controls on the ListControl user interface to instruct the SearchDataSource regarding how to sort the search.

For example, in the Example Order Management app, the NewOrdersDataSource's Sorting property contains the value {Binding RecentNewOrdersList.Sorting}.

2.3.5 Formatted Columns

SPEAK supports limited formatting options for dates in columns.

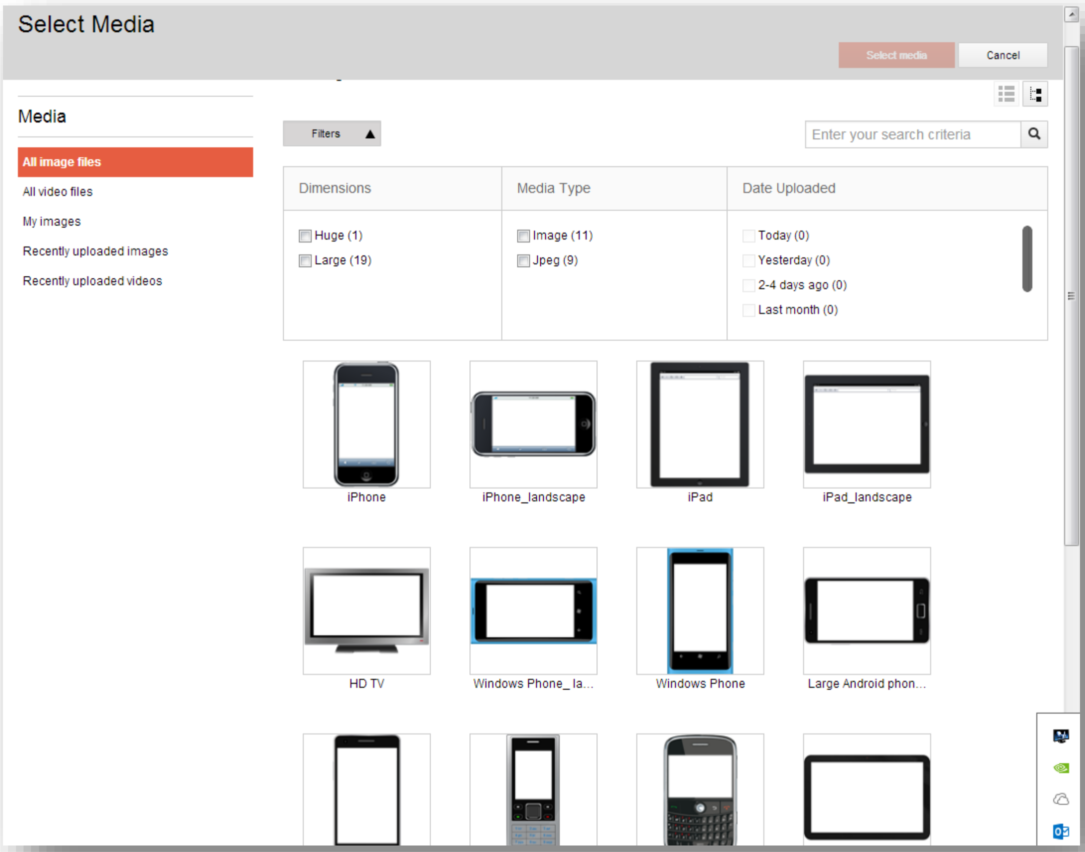
TO DISPLAY	AS	USE THIS FORMAT
Months	1–12	"m"
Months	01–12	"mm"
Days	1–31	"d"
Days	01–31	"dd"
Years	00–99	"yy"
Years	1900–9999	"yyyy"
Hours	0–12	"h"
Hours	00–23	"hh"
Minutes	0–59	"m"
Minutes	00–59	"mm"
Seconds	0–59	"s"
Seconds	00–59	"ss"
Seconds	00.00–59.99	"ss.00"
Time	4 AM	"h AM/PM"
Time	4:36 PM	"h:mm AM/PM"
Time	4:36:03 P	"h:mm:ss A/P"
CMS Short date and time	4/7/2008 1:59 PM	“short”
CMS Long date and time	Monday, April 07, 2008 1:59 PM	“long”

Note: The m or the mm codes are interpreted as minutes when they appear immediately after the h or hh code or immediately before the ss code

Note: Long and short date formats will work only when the 'Formatting' property of the datasource is set to 'add'. Other date formats will work when the 'Formatting' property of the datasource is set to 'add' or is empty.

2.4 Search, Facets and Filters

The Select Media dialog represents a design pattern that is often useful. The dialog looks like this:

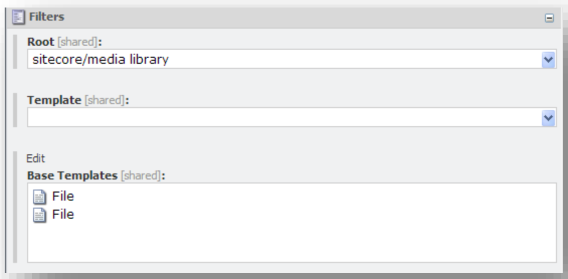


Users have three ways of changing the set of items the dialog shows:

- They can use the links under “Media” to change between seeing “All image files”, “All video files”, “My images”, “Recently uploaded image”, and “Recently updated video”. You can implement functionality similar to this using a HyperlinkButtonsGroup control.
- They can use the search text box to enter a search term. You can implement functionality similar to this by binding a text box to a SearchDatasource.
- They can use the filters (“Dimensions”, “Media Type”, and “Date Uploaded”) to select sets of items where the value of a field divides the set of items into a number of smaller subsets. This is also called “faceted classification” and the groupings (“Dimensions” and so forth) are called “facets.” You can implement functionality similar to this using a FilterControl control.

2.4.1 The HyperlinkButtonsGroup: How and Why

When users open the dialog, they see a list of items specified by a SearchPanel Config item for the SearchDataSource called “AllMediaFiles”. The important configuration is this part:


Filters

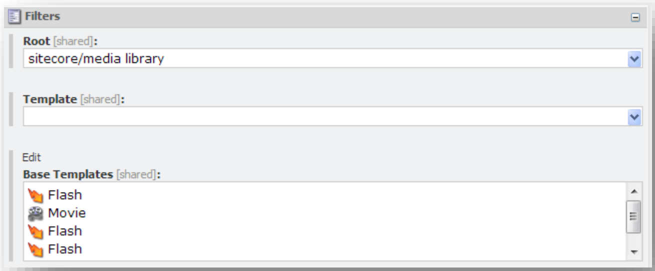
Root [shared]:
sitecore/media library

Template [shared]:

Edit
Base Templates [shared]:
File
File

This specifies that the SearchDataSource retrieves items from the “sitecore/media library” folder and the subfolders of this folder, and that it retrieves items based on one of the two “File” templates (there are two templates because one is “versioned” and the other one is “unversioned”).

When users click the “All video files” hyperlink, the dialog shows only video files. Another configuration item specifies this:


Filters

Root [shared]:
sitecore/media library

Template [shared]:

Edit
Base Templates [shared]:
Flash
Movie
Flash
Flash

This specifies the same Root as before, but changes the Base Templates to the templates you use for items that are videos.

There are also configuration items for “My images”, “Recently uploaded image”, and “Recently updated video”.

The SearchDataSource uses the “All media files” configuration when the dialog opens because this is set in the SearchConfigItemId property.

The four links (“All image files”, “All video files”, “My images”, “Recently uploaded image”, and “Recently updated video”) are all HyperlinkButtons item. They are contained by a HyperlinkButtonGroup control.

The “All video files” HyperlinkButton item has this specification in the Click field:

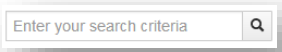
```
set:DataSource({"searchConfig": "{378D67C9-279A-4FBB-AB43-52EC07034FE8}"})
```

When users click this HyperlinkButton item, SPEAK executes this JavaScript. The JavaScript uses the “set” View Member of the SearchDataSource in the dialog to change the value of the “searchConfig” Model Member. This Model Member corresponds to the SearchConfigItemId property. It sets this property to the “AllVideoFiles” configuration item.

The “All media files” HyperlinkButton item is necessary in order to give users a way to go back and see all media files without having to reload the whole dialog.

2.4.2 Creating a Search Text Box

Users can enter text in search text box and either press Return or click the magnifier icon to execute the search:



You implement something similar like this:

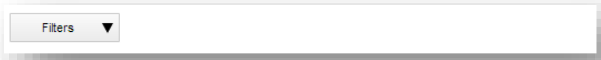
- Add a SearchPanel component to your layout.
  This component adds placeholders that will help you layout and style correctly.
- Add a TextBox control to the Searches placeholder of the SearchPanel.
  Specify a text in the Watermark property. The TextBox will show this text when the Text property is empty. You use it for user assistance, for example: “Enter your search criteria”.
- Add an IconButton control to the Searches placeholder.
  The IconButton in the screenshot uses a magnifier sprite. It has this JavaScript in the Click event:
  `javascript:app.DataSourceId.refresh()`
- The dialog has a SearchDataSource component (“DataSourceId”). The Text property of this SearchDataSource data binds to the Text property of the TextBox with this:
  `{Binding TextBox.Text}`

When a user enters a text in the TextBox and then presses Enter or clicks the IconButton, the SearchDataSource searches for the text entered.

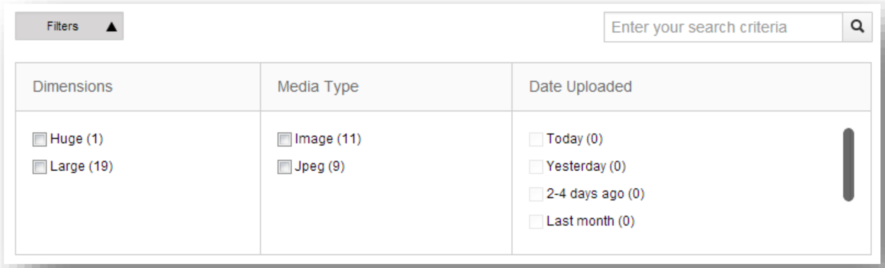
Note that you do not have to add JavaScript for the TextBox: when users press Enter, SPEAK makes sure the firing and wiring of events works. You do have to tell the IconButton what to do when it is clicked because this control does not know when user input is ready in the TextBox until this happens – and the IconButton itself is not connected to the SearchDataSource.

2.4.3 How to use the FilterControl

The FilterControl lets users filter items by *facets.* In the Select Media dialog, the FilterControl is collapsed initially:



When users click the Filters button, the FilterControl becomes visible:



“Dimensions”, “Media Type”, and “Date Uploaded” are *facets.* The Dimensions facet groups items into two sets, “Huge” and “Large”, the Media Type facet group items into “Image” and “Jpeg”, and so forth.

Users can select, for example, “Huge” and the SearchDataSource will only retrieve items in this group. If users also select “Image, the SearchDataSource will retrieve items that belong in both groups (if any such items exist). When users have not selected any group in any facet, the SearchDataSource retrieves all items.

When you have a SearchDataSource component in your layout, you can create the facet functionality in this way:

- Create and configure a Facet item for each facet you need.
- Point the FacetsRootItemId of your SearchDataSoruce to the folder where you have created the Facet items.
- Add a FilterControl component to you layout. Configure the FilterControl and the SearchDataSource to work together.
- The Filters button is optional. If you do not require that users can hide and unhide the FilterControl, you do not need it. If you need it, you add a ToggleButton to the layout.

Each of these steps is described in more detail in the following subsections.

Create and Configure Facet items

Add a folder under your Page Settings for facets. Create an item based on the Facet template for each fact you want, and put these items in the folder you created.

You can add the logic that groups items to the Facet item in two ways:

- You can use the WebAPI. This version of the Cookbook does not include information about this.
- You set the name of a field in the template the items are based on in the FieldName field. Sitecore uses the values in this field for grouping the items. You have to careful: if the field you specify has many different values across items, the facet will not be very useful. It is best to use fields where the values belong to a relatively small set. In the Select Media dialog, for example, the Media Type facet uses the TemplateName field to group items.

Add and Configure the FilterControl

Add a FilterControl control to your layout. Bind the Facets property of this control to the Facets property of your SearchDataSource component.

Configure the SearchDataSource

Bind the SelectedFacets property of the SearchDataSource component to the SelectedFacets property of the FilterControl control.

Add a Button to Open and Close the FilterControl

Add a ToggleButton control to your layout. The Filters placeholder of the SearchPanel control is the preferred placeholder for this button.

Configure the ToggleButton control like this:

- Set the ButtonType property to Default.
- Bind the IsEnabled property to the hasItems member of your SearchDataSource. This tells SPEAK to only show the ToggleButton when the SearchDataSource has > 0 items.
- Set the ShowArrow property to 1. The ToggleButton will show an arrow that indicates what the toggle state of the button is.
- Add the following JavaScript to the Click property of the ToggleButton control:

  ```
  javascript:app.FilterControl.toggle()
  ```

When users click the TogleButton control, this JavaScript will be executed and it will trigger the toggle View Member of the FilterControl.

2.5 Navigating to another Page in an Application

SPEAK pages have a URL just like other web pages. To navigate to a SPEAK page, users simply navigate to the page's URL.

2.5.1 Navigation via Control Configuration

Developers can make this easy by adding a control that navigates to the desired page. Here's a few common controls used for navigation, and the way to configure the control to navigate to a specific URL.

Control	Property	Example Value
• HyperlinkButton	NavigateUrl	/sitecore/Client/Sitecore/Applications/Examples/OM/pages/EditPage
• IconHyperlinkButton	NavigateUrl	/sitecore/Client/Sitecore/Applications/Examples/OM/pages/EditPage
• Button	Click	javascript:window.location.replace('/sitecore/client/sitecore /applications/examples/om/pages/startpage')
• IconButton	Click	javascript:window.location.replace('/sitecore/client/sitecore /applications/examples/om/pages/startpage')

Developers can also add a HyperlinkButtonGroup to provide a group of navigation links. In this case, the developer adds the HyperlinkButtonGroup control to the page and creates a folder of HyperlinkButton Parameter items which specify the location of the destination items in the NavigateUrl field.

2.5.2 Navigation Programmatically

When developers want to modify the URL, for example, to add a query string to it, they can do so programmatically. For example, in the Example Order Management application, the Edit button on the Start Page SmartPanel adds the GUID of the selected item in a ListControl as a query string to the Edit Order page.

The example application developers created the onEditOrder function in the StartPage's JavaScript page code file to append the GUID.

To call the onEditOrder method, the developers provided the following value in the Click property of the Edit Button:

```
javascript:app.onEditOrder();
```

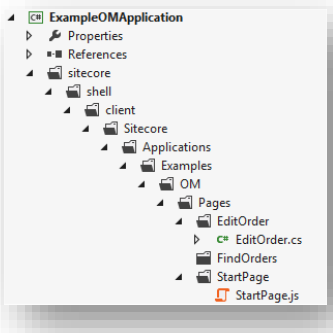
The method has the following definition:

```
onEditOrder: function () {

window.location.replace('/sitecore/client/sitecore/applications/examples/om/Pages/edit
order?o='
                                + this.SelectedGuid.get("text"));
}
```

2.6 Adding Business Logic to a Page with a PageCode Script

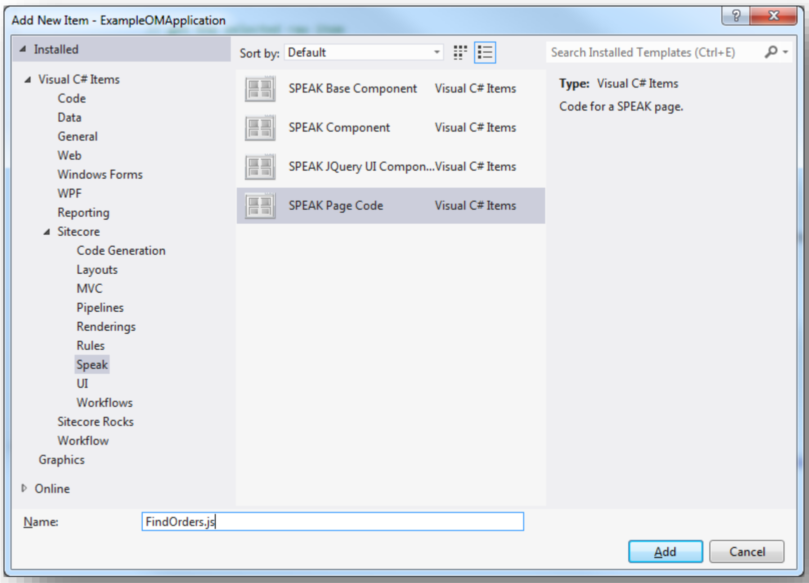
- Open the solution associated with your project (if you haven't created a solution file, do so)
- In the Solution Explorer, create a folder structure that corresponds to the item structure where you have stored your page definition item, including a folder with the same name as the page definition item.
  - For example, the folder structure below matches the Example Order Management application's item structure for the EditOrder, FindOrders, and StartPage.
  - The solution already includes a both a C# and a JavaScript page code file with appropriate names.



- Right click on the Folder that corresponds to the page you want to add page code for and select the Add menu.
  - If you want to create a JavaScript page code file, read the JavaScript Page Code section.
  - If you want to create a C# page code file, read the C# Page Code section.

2.6.1 JavaScript Page Code

- For a JavaScript page code file, choose New Item…
  - Then choose SPEAK Page Code from the dialog and type a name that matches the definition item name.


Add New Item - ExampleOMApplication

Installed

- Visual C# Items
  - Code
  - Data
  - General
  - Web
  - Windows Forms
  - WPF
  - Reporting
  - Sitecore
    - Code Generation
    - Layouts
    - MVC
    - Pipelines
    - Renderings
    - Rules
    - Speak
    - UI
    - Workflows
  - Sitecore Rocks
  - Workflow
- Graphics

Online

Sort by: Default

SPEAK Base Component	Visual C# Items
SPEAK Component	Visual C# Items
SPEAK JQuery UI Compon...	Visual C# Items
SPEAK Page Code	Visual C# Items

Search Installed Templates (Ctrl+E)

Type: Visual C# Items
Code for a SPEAK page.

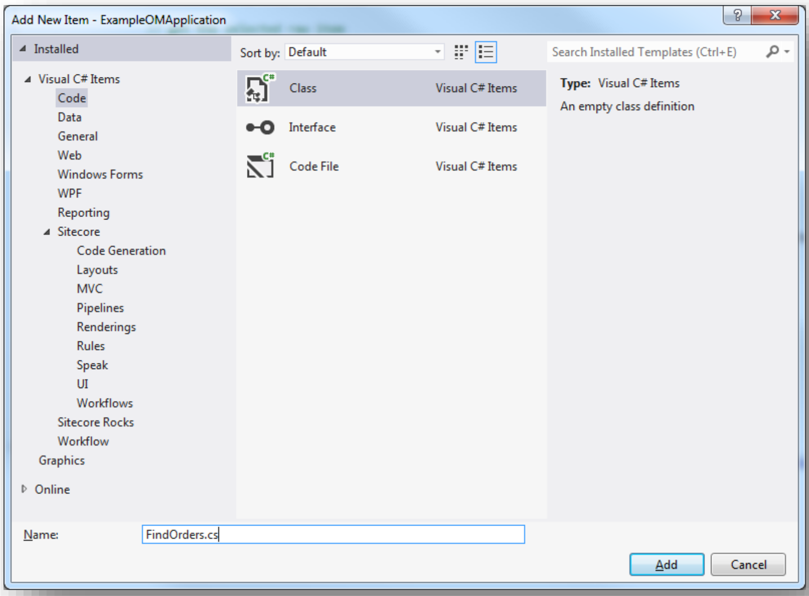
Name: FindOrders.js

Add Cancel

Visual Studio will create a new JavaScript page code script file with the minimal required code. You may then add code to the file to implement business logic.

2.6.2 C# Page Code

- For a C# page code file, choose New Class…
  - Then choose Class from the dialog and type a name that matches the definition item name.


Add New Item - ExampleOMApplication

Installed

- Visual C# Items
  - Code
  - Data
  - General
  - Web
  - Windows Forms
  - WPF
  - Reporting
  - Sitecore
    - Code Generation
    - Layouts
    - MVC
    - Pipelines
    - Renderings
    - Rules
    - Speak
    - UI
    - Workflows
  - Sitecore Rocks
  - Workflow
- Graphics

Online

Sort by: Default

Class	Visual C# Items
Interface	Visual C# Items
Code File	Visual C# Items

Search Installed Templates (Ctrl+E)

Type: Visual C# Items
An empty class definition

Name: FindOrders.cs

Add Cancel

Visual Studio will create a new C# page code script file. Make the class inherit from:

```
Sitecore.Web.PageCodes.PageCodeBase
```

To reference components that you have created on the page, add the following line for each component, using the Id for the control as the name. For example, for a component with Id set to "MyComponent", add the following inside the class definition and before the method definitions:

```
public Sitecore.Mvc.Presentation.Rendering MyComponent { get; set; }
```

Your solution will require references to the following Sitecore libraries:

- Sitecore.Kernel
- Sitecore.Mvc
- Sitecore.Speak.Client

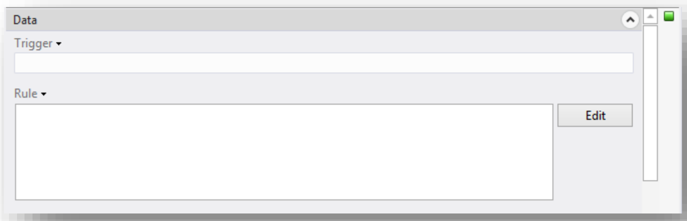
You may then add code to the class that accesses components on the page and implements business logic.

2.7 Adding Business Logic to a Page with Rules

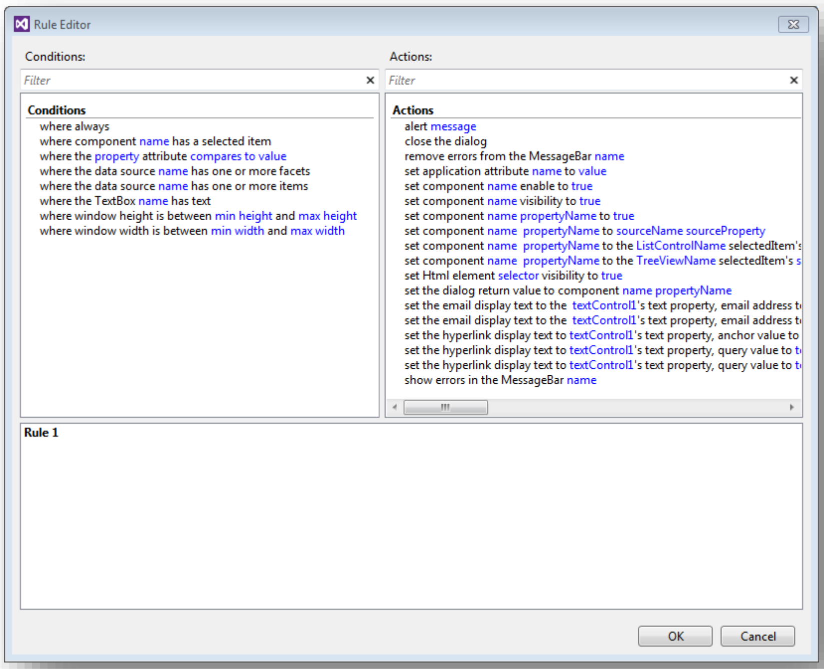
You can also add business logic to a page with rules. A Rule component triggers a rule when another SPEAK component raises an event. The rule itself is in a Rule item that you create.

2.7.1 How to create a Rule item

You create a rule by creating a Rule item, based on the RuleDefinition template, under the Page Settings item for the page where you want to have the rule. Open this Rule item in Sitecore Rocks:



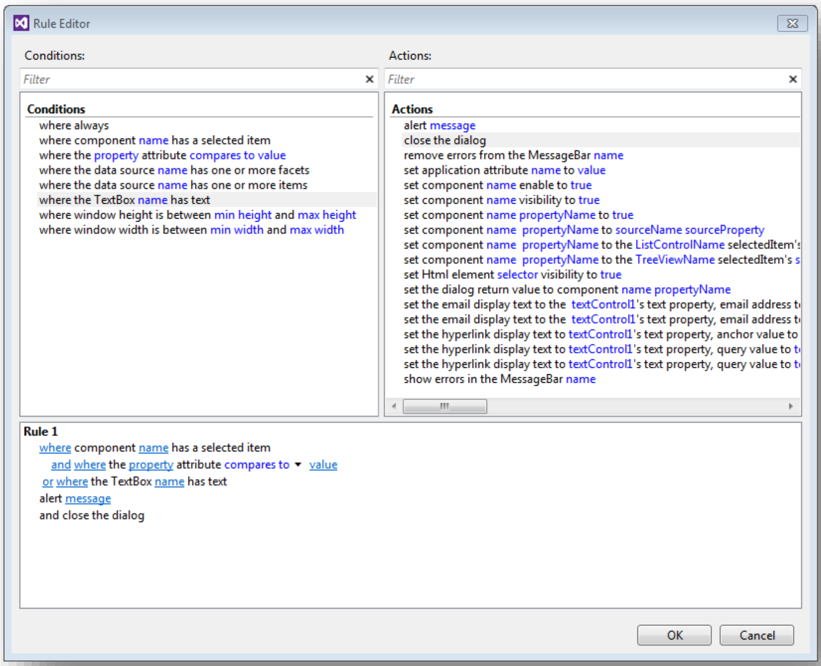
Click the Edit button to open the Rule Editor:



Conditions
- where always
- where component name has a selected item
- where the property attribute compares to value
- where the data source name has one or more facets
- where the data source name has one or more items
- where the TextBox name has text
- where window height is between min height and max height
- where window width is between min width and max width

Actions
- alert message
- close the dialog
- remove errors from the MessageBar name
- set application attribute name to value
- set component name enable to true
- set component name visibility to true
- set component name propertyName to true
- set component name propertyName to sourceName sourceProperty
- set component name propertyName to the ListControlName selectedItem'
- set component name propertyName to the TreeViewName selectedItem's
- set Html element selector visibility to true
- set the dialog return value to component name propertyName
- set the email display text to the textControl1's text property, email address t
- set the email display text to the textControl1's text property, email address t
- set the hyperlink display text to textControl1's text property, anchor value to
- set the hyperlink display text to textControl1's text property, query value to t
- set the hyperlink display text to textControl1's text property, query value to t
- show errors in the MessageBar name

A rule consists of a *condition* and an *action*. You select one or more conditions and actions by double-clicking in the Conditions or Actions panes. When you double-click a condition or an action, it is transferred to the lower pane (the Rule pane):


Rule Editor

Conditions:

Filter

Conditions
- where always
- where component name has a selected item
- where the property attribute compares to value
- where the data source name has one or more facets
- where the data source name has one or more items
- where the TextBox name has text
- where window height is between min height and max height
- where window width is between min width and max width

Actions:

Filter

Actions
- alert message
- close the dialog
- remove errors from the MessageBar name
- set application attribute name to value
- set component name enable to true
- set component name visibility to true
- set component name propertyName to true
- set component name propertyName to sourceName sourceProperty
- set component name propertyName to the ListControlName selectedItem'
- set component name propertyName to the TreeViewName selectedItem's
- set Html element selector visibility to true
- set the dialog return value to component name propertyName
- set the email display text to the textControl1's text property, email address
- set the email display text to the textControl1's text property, email address
- set the hyperlink display text to textControl1's text property, anchor value to
- set the hyperlink display text to textControl1's text property, query value to
- set the hyperlink display text to textControl1's text property, query value to
- show errors in the MessageBar name

Rule 1

where component name has a selected item
and where the property attribute compares to value
or where the TextBox name has text
alert message
and close the dialog

OK Cancel

Note the following:

- You can specify multiple conditions and actions for a rule.
- You can specify that conditions are “and’ed” or “or’ed” together. You click the “and” to change it to “or”, and vice versa.
- Sitecore executes all actions of a rule when the conditions of the rule evaluate to True.
- Conditions and actions have placeholders. The placeholders are blue in the Rule pane. Some placeholders are drop-downs. You must replace placeholders with actual data by clicking the placeholder and selecting or entering the value you want.
- You can create more than one rule. The Rule Editor adds conditions and actions to the current selected rule by default. You can create a new rule by right-clicking and selecting Add New Rule from the context menu. Sitecore executes the rules sequentially.
- You can delete a rule by condition, action, or rule by right-clicking and selecting Delete.

2.7.2 How to configure a Rule component

You connect the Rule component to a Rule item in the RuleItemId property of the Rule component. Follow these steps:

- Add a Rule component to the page in the Layout Designer.
- Specify the RuleDefinition item you have created in the RuleItemID property.
- Specify a control in your page in the TargetControl property. SPEAK will listen for events that are raised by this control.
- Specify an event in the Trigger property. SPEAK will listen for this event.

If you do not specify a control in TargetControl, SPEAK will listen for events raised by the “window” default component.

You add a Rule component to your page for each event you are interested in, and a page can have multiple Rule components. Several different Rule components can point to the same Rule item.

Chapter 3

Q&A

This chapter contains a series of frequently asked questions and answers.

3.1 Can I use the Content Editor, Page Editor, and other Sitecore client tools to edit my SPEAK application?

The traditional Sitecore clients do not support all aspects of editing SPEAK applications yet. For now, developers must use Sitecore Rocks to edit SPEAK applications.

3.2 What is a definition item?

Nearly all the entities that you create in an application, such as pages, controls, and data templates, have associated properties. In Sitecore and in SPEAK, developers create an item to define these properties. This document refers to such an item as a “definition item”.

For example, to create a page, a developer creates a page definition item and configures that item with the information that describes the behavior of the page. To see the page in a browser, a user navigates to the page definition item. Sitecore translates the definition item into the desired page and behavior.

In most cases, one definition item will refer to many other definition items to give the complete description of the desired behavior.

3.3 How should I organize my application's definition items?

SPEAK applications typically require many definition items. Defining an information architecture with standard locations for all types of definition items early will help developers quickly decide where to store and locate definition items. The complexity of the information architecture required depends on the complexity of the application.

A simple application with a relatively small number of pages may use an information architecture similar to the one shown below:

- <Application Folder>
  - <Pages Folder>
    - <Page 1 Definition Item>
      - Page Settings
        - <Component 1 Configuration Item>
        - <Component 2 Configuration Item>
    - <Page 2 Definition Item>
      - Page Settings
        - <Component 1 Configuration Item>
        - <Component 2 Configuration Item>
  - <Shared Settings Folder>
    - <Common Component 1 Configuration Item>
    - <Common Component 2 Configuration Item>
  - <Templates>
    - <X entity template>
    - <Y entity template>
  - <Test Data Folder>
    - <X Entities>
      - <X entity 1>
      - <X entity 2>
    - <Y Entities>
      - <Y entity 1>
      - <Y entity 2>

3.4 What is a “configuration item”?

All components have properties that you can set in the Visual Studio properties panel. Some controls, however, require more properties for proper configuration than can appear in the properties panel. In such cases, developers create one or more configuration items which define the properties of the control. By convention, developers create configuration items underneath the PageSettings item, which is a direct subitem of the page definition item.

Developers associate a control with the configuration item by providing the configuration item’s GUID in the control’s DataSource property on the Rocks Layout Designer.

3.4.1 Why should developers define all end-user visible text in configuration items?

For applications that support multiple languages, developers should define all end-user visible text in configuration items for the corresponding controls. This will allow the development team to use standard Sitecore tools for translating text from the original language to all additionally supported languages.

3.4.2 Which template should I use when I create a “configuration item”?

Each control has its own template or group of templates that define the fields required to define the properties of the control. Such templates are usually created underneath the control definition item.

3.5 What types of pages does SPEAK provide and how are they different?

SPEAK defines three types of standard pages:

- Dashboard
- List
- Task

A **Dashboard** page shows overview information for the entire application or a specific area of the application and usually provides navigation to more specific information available in other pages. The dashboard may show a list of entities that meet certain criteria, such as the “best performing” or “worst performing” entities of a certain type. The dashboard page may also show charts and other summary data. Dashboard pages generally do NOT provide any editing capabilities.

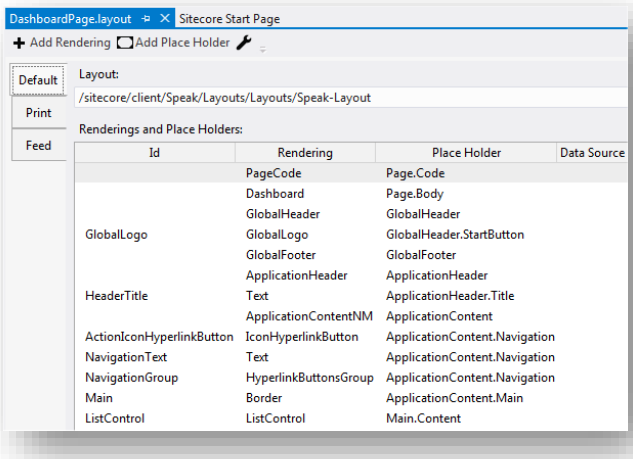
A **List** page helps users locate a specific entity of a given type and provides navigation to more specific information available in other pages. List pages generally display a single large list of entities of a certain type, with search and filtering capabilities provided to help the user locate one or more entities. List pages generally do NOT provide editing capabilities, although some bulk operations could be appropriate.

A **Task** page generally shows detailed information about a single entity, and potentially its related sub-entities. Often task pages provide editing capabilities. Task pages usually do not provide many navigation options, other than the ability to navigate back to the previous page.

3.6 What do the page type branches provide by default?

All the SPEAK page type branches include a number of common features and make it easier to get started when creating a new page. Using the branches will save you time, since you will not need to set up these common features each time you create a page.

Each of the branches includes a page definition item and a Page Settings subitem. The layout definition for the page branches comes pre-configured to use the Speak-Layout and a number of components, as seen in the screen shot below.

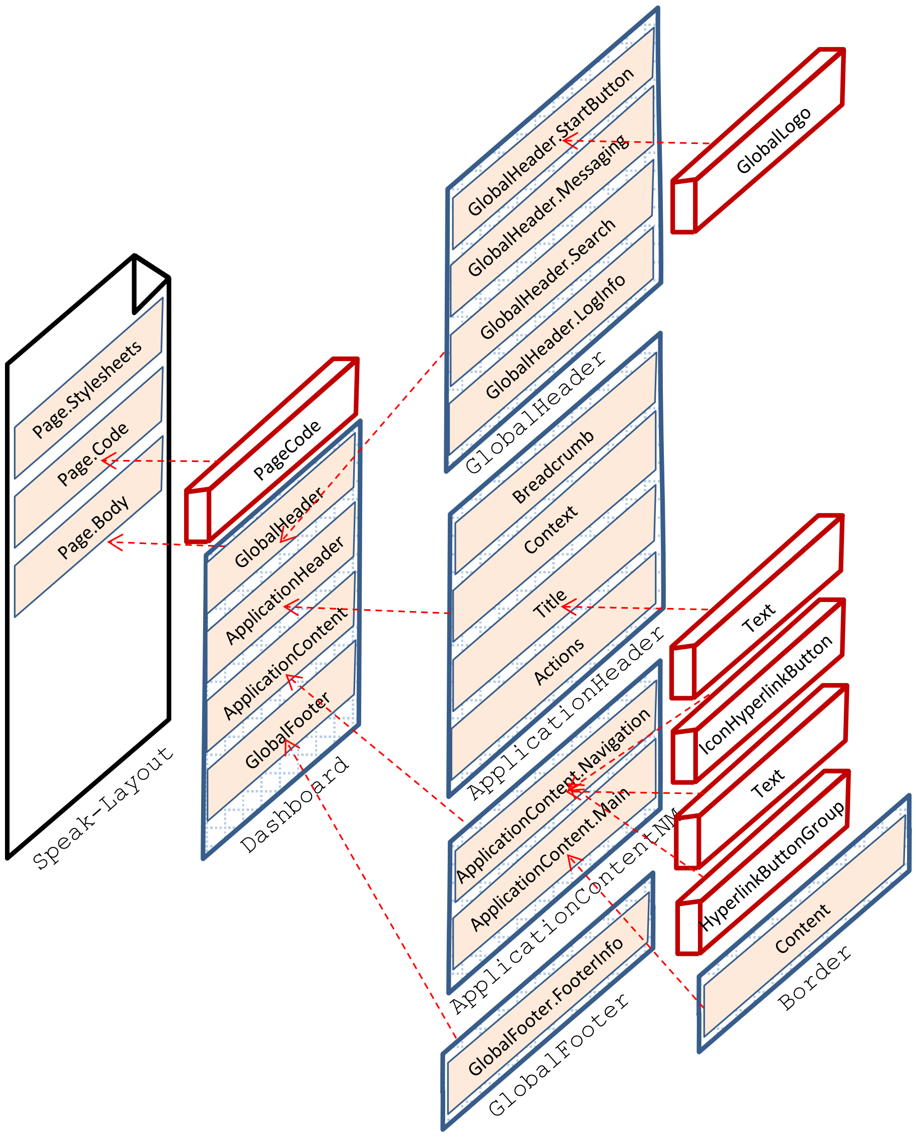


Layout:
/sitecore/client/Speak/Layouts/Layouts/Speak-Layout

Renderings and Place Holders:

Id	Rendering	Place Holder	Data Source
	PageCode	Page.Code	
	Dashboard	Page.Body	
	GlobalHeader	GlobalHeader	
GlobalLogo	GlobalLogo	GlobalHeader.StartButton	
	GlobalFooter	GlobalFooter	
	ApplicationHeader	ApplicationHeader	
HeaderTitle	Text	ApplicationHeader.Title	
	ApplicationContentNM	ApplicationContent	
ActionIconHyperlinkButton	IconHyperlinkButton	ApplicationContent.Navigation	
NavigationText	Text	ApplicationContent.Navigation	
NavigationGroup	HyperlinkButtonsGroup	ApplicationContent.Navigation	
Main	Border	ApplicationContent.Main	
ListControl	ListControl	Main.Content	

The diagram below provides an overview of the layout and components included in the Dashboard page branch.



The Speak-Layout defines three placeholders:

- Page.Stylesheets
- Page.Code
- Page.Body

The default page type branches do not use the Page.Stylesheets placeholder.

The default page type branches all include a PageCode component assigned to the Page.Code placeholder. SPEAK requires that all pages include a PageCode component. The PageCode component implements the default business behavior of the page, runs any rules defined on the page, and calls custom page code scripts (if any are configured).

Each of the page type branches adds a page structure component assigned to the Page.Body placeholder.

- Dashboard pages include the Dashboard page structure component.
- List pages include the List page structure component.
- Task pages include the Task page structure component.

Each of the page structure components defines the following placeholders:

- GlobalHeader
- ApplicationHeader
- ApplicationContent
- GlobalFooter

All three page branches populate the page with appropriate components for each of these placeholders, including:

- The GlobalHeader component assigned to the GlobalHeader placeholder.
- The GlobalLogo control assigned to the GlobalHeader.StartButton placeholder (introduced by the GlobalHeader component).
- The GlobalFooter component assigned to the GlobalFooter placeholder.
- The ApplicationHeader component assigned to the ApplicationHeader placeholder.
- A Text control assigned to the ApplicationHeader.Title placeholder (introduced by the ApplicationHeader component).

Aside from the components listed above, the Dashboard, List, and Task page branches also add some additional components, although the Task page branch adds different components than the Dashboard and List page branches.

The Dashboard and List page branches add the following components:

- The ApplicationContentNM substructure component assigned to the ApplicationContent placeholder.
  - This defines two areas in the content portion of the page, for navigation links and main content.
- An IconHyperlinkButton control assigned to the ApplicationContent.Navigation placeholder (introduced by the ApplicationContentNM component).
  - This is intended as an example navigation button.
- A Text control assigned to the ApplicationContent.Navigation placeholder.
  - This provides a title for an additional group of navigation buttons.
- A HyperlinkButtonsGroup control assigned to the ApplicationContent.Navigation placeholder.
  - This provides an additional group of navigation buttons (empty by default).
- A Border control assigned to the ApplicationContent.Main placeholder.
  - This acts as a container for the main content of the page.

The Task page branch adds the following components:

- The ApplicationContentMI substructure component assigned to the ApplicationContent placeholder.

  - This defines two areas in the content portion of the page, for main content and an information sidebar.
- A Border control assigned to the ApplicationContent.Main placeholder (introduced by the ApplicationContentMI component).
  - This acts as a container for the main content of the page.
- A RowPanel component, named “Buttons”, assigned to the Main.Content placeholder (introduced by the “Main” Border component).
- A ColumnPanel component, named “LeftButtons”, assigned to the Buttons.Content placeholder (introduced by the “Buttons” RowPanel component).
  - This component has its “GridColumns” parameter set to 6, which means it will take up half of the space available in its parent row.
- A ColumnPanel component, named “RightButtons”, assigned to the Buttons.Content placeholder (introduced by the “Buttons” RowPanel component).
  - This component also has its “GridColumns” parameter set to 6, which means it will take up the other half of the space available in its parent row.
  - The GridColumns of all children of a RowPanel should add up to 12.
- A Button control, named “BackButton”, assigned to the LeftButtons.Content placeholder (introduced by the “LeftButtons” ColumnPanel component).
- A Button control, named “SaveButton”, assigned to the RightButtons.Content placeholder (introduced by the “RightButtons” ColumnPanel component).
- A Text control, assigned to the ApplicationContent.Info placeholder (introduced by the ApplicationContentMI component).

3.7 Can I copy an existing SPEAK page to create a new page?

When you have a page with many components, and you want to create a similar page, you may feel tempted to create a copy of the page you already have.

Unfortunately, this won’t work.

Remember that components often reference configuration items by including the GUID of the configuration item in the component properties.

When you copy the SPEAK page definition item, you will get the list of components in the new copy, and the properties for all the components will be the same as the original properties, including the GUIDs which point to the component property items.

This means that the copy will still point to the original configuration items, even though you have made copies of the original configuration items.

Of course, you could then manually change the GUIDs to the new configuration item copies, but most developers consider this a tedious task.

3.8 What is a SPEAK placeholder, where does it come from, and how is it used?

A placeholder provides a named location in a page.

SPEAK provides several placeholders by default in the Speak-Layout and a number of components add placeholders to a page when they are assigned to the page. When developers add a component to a page, they assign a placeholder key to the component, and this indicates where on the page SPEAK will position the component.

Some components have a placeholder key predefined, so developers do not need to provide the placeholder key themselves.

3.9 What is the difference between a “component” and a “control”?

A SPEAK component is any Sitecore rendering that developers can assign to a SPEAK page. SPEAK components do not necessarily have a visual representation on the SPEAK page. A component may, however, add placeholders to a page.

SPEAK controls are components that do have a visual representation on the page. The difference is mainly one of convention. By convention, we call components that have a visual representation a control.

For example, Button is both a SPEAK component and a SPEAK control, while PageCode is a SPEAK component and is not a SPEAK control.

3.10 What do N, M, and I stand for in the substructure component names?

SPEAK defines a number of substructure components with names like:

- ApplicationContentM
- ApplicationContentMI
- ApplicationContentNM
- ApplicationContentNMI
- DialogContentMI
- DialogContentNM

In these components, the letters N, M, and I stand for:

- Navigation panel (shown on the left side of the page)
- Main content area (to the right of the navigation panel and to the left of the information panel)
- Information panel (shown on the right side of the page)

3.11 How do I open the Layout Designer in Sitecore Rocks?

1. Open Visual Studio with Rocks installed.
2. Locate a SPEAK Page and right click on it.
3. Select the Tasks > Design Layout menu option (or type Ctrl + Shift + U).

3.12 What can I specify in the Click property?

Some components provide interactions. For example, the Button control has a property called “Click”. You use this property to specify what happens when a user clicks the button.

You can specify the interaction using four different protocols:

Protocol	Description
click	The field contains a JavaScript function that SPEAK executes when the interaction is invoked.
command	The field contains a JavaScript command that SPEAK executes when the interaction is invoked.
serverclick	The field contains an MVC controller action that is called when the interaction is invoked. The format is: “controller name/action”. The controller name does not need the Controller postfix.
trigger	The field contains a trigger action.

The default protocol is “trigger”, but you can overwrite this by prefixing the field with “javascript:”, “command:” or “serverclick:”.

For example, to show an alert box, you set the Click property set to:

```
javascript: alert(‘Hello’)
```

If a component has the ID “item” and this component has a member called “save”, you can trigger “`item:save`” by specifying either “`item:save`” or “`trigger:item:save`” in the Click property.

You can trigger events on other components in the same SPEAK page as the component that has the user interaction. For example, you can specify that when a user clicks a button, another control become visible.

3.13 What Syntax Can I Use When Data Binding?

You can data bind a property to a property of another property in different ways. This table shows the ways you can specify the value of a property:

Syntax	Description
`Hello World`	Literal value: Hello World
`{Binding Name=Button1.Text}`	Binding to Button1.Text
`{Binding Button1.Text}`	Equivalent to {Binding Name=Button1.Text}
`{Binding ListControl1.Header.Text}`	Binding to ListControl1.Header.Text
`\{Hello World\}`	Literal value: {Hello World}
`{Binding Button1.IsVisible, Converter=Not}`	Binding to Button1.IsVisible using converter function Not
`{Binding Button1.Text, DefaultValue=Denmark}`	Binding to Button1.Text, setting the default value to the literal Denmark.
`{Binding Button1.Text, DefaultValue={XPath @SampleField}}`	Binding to Button1.Text, setting the default value to result of the XPath expression. This allows for very advanced initialization of the value.
`{Binding Name={XPath @BindToField}}`	Binding to the value of the BindToField in the current data source.

Note that you can, for example, bind a property that takes True or False as possible values to a property of another control that can also be True or False. You can use this to create various interactions, like this:

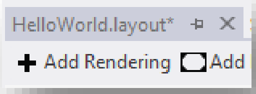
Bind the IsVisible property of a control to the IsChecked property of a CheckBox. When a user selects the CheckBox, the other control becomes visible; when the user deselects the CheckBox, the other control becomes invisible.

Many property fields have a dropdown in Sitecore Rocks, and SPEAK often populates this dropdown with bindings. You can always type in another binding that is not in the dropdown list manually.

3.14 How Do I Get Sitecore Rocks to Save?

You can sometimes encounter a slight problem when saving in Sitecore Rocks and Visual Studio: Visual Studio does not realize that you have modified a component in a SPEAK page and the page is now “dirty” and needs to be saved. You can also sometimes see that there is a slight delay after you have modified a component and Visual Studio realizes that the page is “dirty”. If you try to save the page and Visual Studio does not consider the page to be dirty, it will not save anything.

You should always check that Visual Studio is ready to save a page. It will put an asterisk in the tab in the editor when it considers a page “dirty”:



If Visual Studio does not recognize that you have modified something in a page, try this:

- Select a property and enter some text. Click in another property, and then return and delete the text you just entered.
- It sometimes helps if you only have one tab open.

Chapter 4

Order Management Application

The SPEAK Team provides a small example Order Management application which offers a useful self-study tool for development teams. The application provides working examples of many components. This chapter provides information about the application.

4.1 Application Scenario

This example application provides simplified, real world examples of typical user interface constructs as a self-study teaching tool for developers learning SPEAK.

The application simulates a simple order management application for a small shop. It implements just three pages:

- **The start page** – provides an overview of new and in progress orders. Selecting an order opens a "smart panel" on the right side of the browser, which provides more information about the order and a button which links to the Edit Order page.
  The start page is an example of a SPEAK dashboard page.

- **A find orders page** – provides searching and filtering to locate a specific order. Selecting an order enables a button which links to the Edit Order page (not implemented).
  The find orders page is an example of a SPEAK list page.

- **An edit order page** – provides editing features for modifying orders.
  The edit order page is an example of a SPEAK task page.

The application provides examples of how to configure and use many commonly used SPEAK components, including:

• Accordion	on the Edit Order page, in the Order Details tab
• Border	on all three pages
• Button	on all three pages
• ColumnPanel	on all three pages
• FilterControl	on the Find Orders page
• HyperlinkButtonGroup	on the Start Page and Find Orders page
• IconButton	on the Find Orders page
• IconHyperlinkButtton	on all three pages
• ListControl	on all three pages
• RadioButton	on the Edit Order page, in the Shipping tab
• RowPanel	on all three pages
• Search	on the Find Orders page
• SearchDataSource	on all three pages
• SmartPanel	on the Start Page
• TabControl	on the Edit Order page
• Text	on all three pages

4.2 How does the application retrieve data?

All applications must retrieve stored data and display it to users. The Order Management example application simulates data storage using a “Fake Data” folder in the Sitecore content tree. The folder contains items which simulate orders, and sub-items which simulate order line items. A real world application would most likely choose a different strategy for storing information.

As an example, however, this approach demonstrates how to retrieve data and display it in a SPEAK control. The application uses several SearchDataSource components, which retrieve Sitecore items from the content tree, starting at a specific location on the tree and based on various search criteria associated with the data source component.

In a more complex, real world application, developers typically create a controller mechanism that allows the front end to request data from the backend.

4.2.1 How do the two ListControls on the StartPage filter Orders?

The Start Page shows two ListControls, one which displays only “New” orders (RecentNewOrdersList), the other which displays only “In Progress” orders (InProgressOrdersList).

Each of these ListControls has an Items property which is bound to a SearchDataSource control’s Items property (NewOrdersDataSource and InProgressOrdersDataSource respectively).

Each SearchDataSource has a SearchConfigItemId property which contains the GUID of an item based on the SearchPanel Config template (stored under the …/StartPage/PageSettings item, the items are named “New Orders” and “In Progress Orders” respectively).

Each of these items includes a Search field. The Search field accepts filters that have the format:

```
(<field name>:<field value>)
```

The fake data order items include a field named Status of the field type DropLink. DropLink displays values provided under the Core:/sitecore/Client/Sitecore/Applications/Examples/OM/Fake Data/Status folder. The values include New, In Progress, Closed, and Cancelled, each represented by an item.

To filter the ListControl to only show items that have a specific value in the Status field (for example, “New”), the developer has placed the following filter in the Search field of the “New Orders” SearchPanel Config item:

```
(Status:ab00415cde42470eb9331f1e089b757c)
```

Where “`ab00415cde42470eb9331f1e089b757c`” is the GUID of the “New” item displayed in the Status DropLink field.