



Sitecore Intranet Portal 3.2 Developer's Guide

The Quick Start Guide and Configuration Reference for Administrators and Developers

Table of Contents

Table of Contents.....	2
Chapter 1 Sitecore Intranet Portal Administrator's Default Credentials.....	5
Chapter 2 Site Config Settings.....	6
2.1 Configuration Items.....	7
2.1.1 Settings.....	7
2.1.2 Locations.....	8
2.1.3 Portal.....	8
2.2 System Folders.....	10
Chapter 3 User Administration.....	11
3.1 Profiles.....	12
3.1.1 Automatically Creating and Updating User Profiles.....	12
3.1.2 Manually Creating User Profiles.....	12
3.2 Setting Security for User Profiles.....	14
3.2.1 Default Security Settings.....	14
3.2.2 Resetting Security Permissions for every User Profile.....	14
3.2.3 Main Sitecore Intranet Portal Roles.....	14
3.3 Editing User Profiles.....	16
3.3.1 Allowing Employees to Edit their User Profiles.....	16
3.3.2 Adding Fields to User Profiles.....	17
Adding Fields to Intranet.Employee.xslt.....	17
Making Fields Searchable in the Phonebook and Pager.....	17
3.4 Extending the Portal.....	18
3.4.1 Active Directory Integration.....	18
3.4.2 Setting Location, Department and Canteen Automatically.....	18
Chapter 4 Skinning.....	19
4.1 Skins.....	20
4.1.1 Changing Skins.....	20
4.1.2 Department and Location Based Skins.....	21
4.1.3 Customizing Icons and Images.....	21
4.1.4 Changing the Logo.....	22
4.1.5 Changing the Top Right-hand Picture.....	23
4.1.6 Customizing the Icons at the Top of the Page.....	23
4.1.7 Customizing the Login Page.....	24
4.1.8 Creating a Custom Skin.....	25
4.2 Customizing Style Sheets.....	26
4.2.1 Referencing Skin Images from CSS Files.....	29
4.3 Developer Hints.....	30
4.3.1 Dynamic Image Scaling.....	30
4.3.2 Using Skinned Images in XSLT Files.....	30
Chapter 5 Web Parts.....	31
5.1 Web Parts Configuration.....	32
5.1.1 Security and Rights Management.....	33
5.2 Standard Web Parts.....	34
5.2.1 RSS Web Part.....	35
5.3 Web Parts, Sublayouts and Web Controls.....	36
5.3.1 Example: Using a sublayout as a Web part.....	36
5.3.2 Example: Creating Web Parts Based on the Web Part Template.....	37
5.4 Extending the Portal. Creating Your Own Web Parts.....	39
5.5 Extending the Portal. Using SharePoint Web Parts.....	42
Chapter 6 Search.....	43
6.1 Configuring Indexing for PDF Files and Office Documents.....	44
6.2 Customizing the Way Search Results are Displayed.....	45
Chapter 7 Editing Using the Page Editor.....	46
7.1 Creating and Editing Content.....	47

7.1.1	Editing Additional Fields	48
7.2	Managing Permissions for Page Editor Features	50
Chapter 8	Multilanguage Features	51
8.1	Layered Translations	52
8.2	Content Area vs. Menu Area	53
8.2.1	Language Behavior in the Menu Area	53
8.2.2	Language Behaviour in the Content Area	53
8.3	Working with Different Languages	55
8.3.1	The Language Selector	55
8.3.2	Editing Multilanguage Content	55
8.3.3	Corporate Language and CorporateTitle field	56
	Changing the Corporate Language	57
8.3.4	Adding a CorporateTitle Field to Custom Templates	57
8.3.5	Toggle Visibility of CorporateTitle for Specific Templates	57
8.4	Working with the Localized Placeholder	58
8.4.1	Using the LanguageResolver Outside the Localized Placeholder	58
8.5	Adding and Removing Languages	59
8.5.1	Adding a New Language	59
8.5.2	Removing the English Language	59
8.5.3	Removing the Language Selectors	59
8.6	Security and Languages	60
Chapter 9	Data Channeling	61
9.1	Target Groups	62
9.1.1	Setting Security for Target Groups	62
9.1.2	Tagging Content with Target Groups	62
9.1.3	Setting Up Templates to Use the Data Channel Framework	63
9.2	Filtering	64
9.2.1	Adding/Removing the Filtering Control on Lists	64
9.2.2	Using DCF for Custom Renderings and Sublayouts	64
9.3	How to Disable the Data Channel Framework	66
Chapter 10	File Drop Area	67
10.1	Attachments and the File Drop Area	68
10.1.1	Enabling the File Drop Area Field	68
10.1.2	Modifying the Views	68
10.1.3	Changing the Drag and Drop Window	68
10.1.4	The Fallbackview Parameter	69
Chapter 11	RSS Feeds	70
11.1	Disabling RSS Feeds on an Intranet Site	71
11.2	Modifying Intranet Portal RSS Feeds	72
11.2.1	The Feed Entry Content Field	72
11.3	Content Item Feeds	74
11.3.1	Subscribing to a Content Item RSS Feed	74
	Using the Content Item RSS Feed	75
Chapter 12	Draft Mode	76
12.1	Draft Versions	77
12.1.1	The Draft Workflow	77
12.1.2	Who is the Author of the Draft Version	77
12.2	Enabling Draft Mode	78
12.3	Creating a Draft Workflow	79
Chapter 13	Other Features	80
13.1	Converting between the Rich Text and Word Field Types	81
Chapter 14	Developer's Notes	82
14.1	Dates	83
14.1.1	Automatically Setting Date or Datetime Fields to Today's Date or Current Timestamp	83
14.1.2	Date Formatting and Parsing	83
14.1.3	Changing Date Format	83
14.2	XSLT	84
14.2.1	Methods to Retrieve the Most Important Items in XSLT	84

- 14.2.2 XSLT Controls and Extensions 84
 - Isfieldvisible 84
 - Longdate 84
 - Shortdate..... 84
- 14.3 Styles..... 85
 - 14.3.1 Word Field Styles 85
- 14.4 Search..... 86
 - 14.4.1 Customize Search Output 86
- 14.5 Draft Mode..... 87
 - 14.5.1 IntranetItemProvider Class..... 87
 - 14.5.2 DraftModeDisabler Class 87
- 14.6 Side Menu 88
 - 14.6.1 Adding Side Menu Item 88

Chapter 1

Sitecore Intranet Portal Administrator's Default Credentials

As an administrator or developer you will work with various Sitecore Intranet Portal (SIP) installations. Here are the default administrator's credentials:

SIP installation	User Name	Password
Demo site (Norman Furniture)	admin	b
Clean SIP 3.1	admin	b

Chapter 2

Site Config Settings

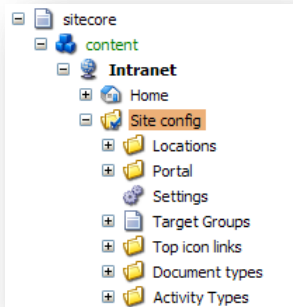
This chapter describes the *Site Config* configuration item and the configuration options it provides including the Frontpage portal setup, Target Groups, Activity Types and the Skin setup.

This chapter contains the following sections:

- Configuration Items
- System Folders

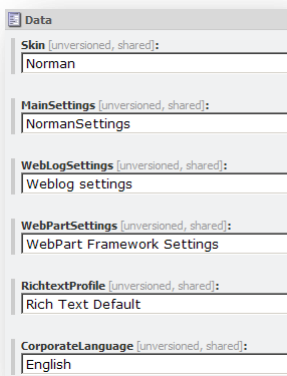
2.1 Configuration Items

This section describes three configuration items: *Settings*, *Locations* and *Portal*. These items are located under the *Site config* item:



2.1.1 Settings

The *Settings* item allows you to switch quickly between different presets. It contains the following settings:



Skin

The “skin” of the site defines how it looks. For more information about skins in Sitecore Intranet Portal, see *Chapter 4, Skinning*. The Skin settings are described in detail in the separate SIP Skin Settings document.

MainSettings

This field contains the definition of the main settings. Setting are located under the `/system/IntranetSettings/MainSettings/` item. The main settings are described in detail in the separate SIP Main Settings document. To read this document, visit the [Sitecore Developers Network](#).

WeblogSettings

The global settings for blogs include date/time format, notification sender email, button titles and so on. The *Weblog settings* are stored under the `/system/IntranetSettings/WeblogSettings/` item.

WebPartSettings

Portal page settings are stored in the *WebPart Framework Settings* item found under the `/system/IntranetSettings/WebPartSettings/` item.

RichTextProfile

The rich text editor profiles are located under the `/system/IntranetSettings/Rich Text Profiles/` item.

Corporate Language

This item defines the corporate language of the Intranet Portal. For more information about this, see *Chapter 8, Multilanguage Features*.

2.1.2 Locations

The *Location* items are used to identify where an employee is located. This is useful if the company's employees are spread among different geographical locations.



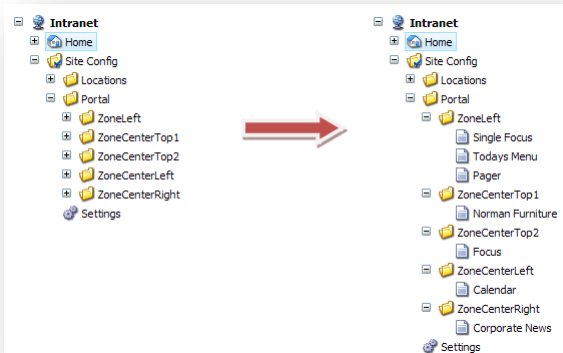
User locations can be specified in intranet user profiles.

All the users associated with a location automatically use the skin assigned to that location, unless a skin has also been defined for the user's department in which case the department skin takes precedence.

2.1.3 Portal

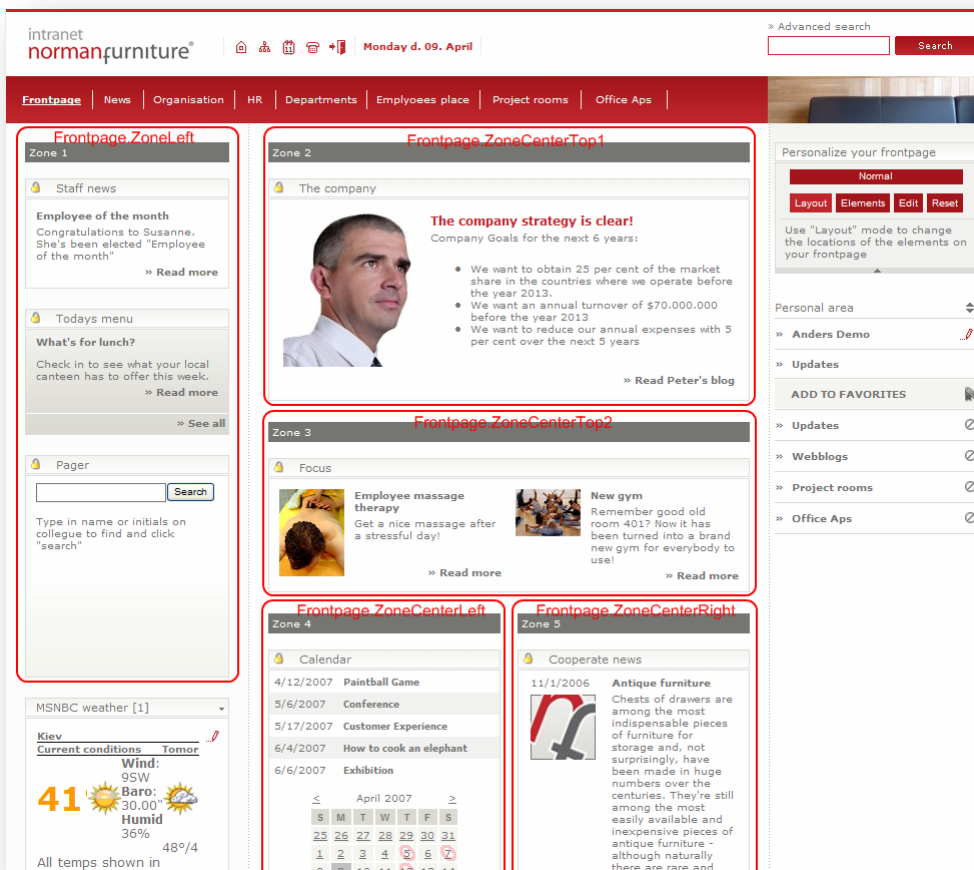
The Sitecore Intranet interface can be customized by the end user, but sometimes administrators require that particular elements cannot be changed and are always visible. Such elements may contain important corporate news or personal task lists. The *Portal* folder allows you to specify the elements which cannot be changed by the Page Editor.

The *Portal* folder contains Web parts which are shown on the front page permanently. In our example the front page is split into five zones, each containing one or more Web parts.



The Web parts from the portal folder cannot be removed by a user from the Page Editor, but if the user has **Write** permissions for the Web part, they can close it. If the user doesn't have *Write* permission, the Web part cannot be closed or moved around on the page. Such parts are marked with

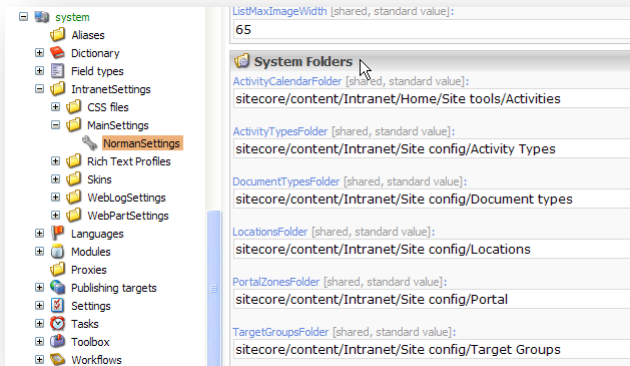
lock icons in the Layout mode.



For more information about the *Portal settings* item and the Web parts it contains, see *Chapter 5, Web Parts*.

2.2 System Folders

The **System Folders** section of the *Main Settings* item allows you to specify different locations for user profiles, the activity calendar, top icon links, activity types, locations and so on.



To get a reference to a system folder from an XSLT file, use the `GetSystemFolder(folderName)` method in the `IntranetUtil` class. See `Intranet.Top.xslt` file for an example.

Chapter 3

User Administration

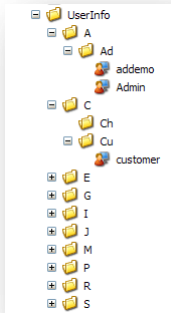
This chapter describes how administrators can manage the users. It also explains how users can edit their own profile information.

This chapter contains the following sections:

- Profiles
- Setting Security for User Profiles
- Editing User Profiles
- Extending the Portal

3.1 Profiles

Users in Sitecore Intranet Portal have profiles that allow them to personalize the portal for themselves. Profiles are stored under the `/Intranet/Home/UserInfo/` item and are by default organized into a hierarchy with 2 levels:



As you can see the hierarchy is created using the first 2 letters of the username as folder names. You can specify the number of folders that are created by changing the `UserProfileTreeDepth` setting in the `web.config` file.

In Sitecore Intranet Portal you use the **User Manager** to create users. A profile for the newly created user is created automatically when the user logs in to Intranet Portal for the first time. By default, profiles are not created automatically for users who are members of the Sitecore domain. This feature can be modified in the `web.config` file, using the `excludeDomains` setting of the `userProfileProvider`. If there are more than one domain, enter them in a comma separated list.

You can also create the profile manually. The name of the user profile item must correspond to the user's username.

Note

Security permissions should be granted or denied to the Sitecore user, not to the Intranet Portal user profile.

3.1.1 Automatically Creating and Updating User Profiles

Many intranet solutions are customized to replicate user profile information within Sitecore from external sources such as Active Directory. In this case, the custom replication code will typically update all the relevant fields in a user profile with the information from Active Directory, while the intranet-specific fields are maintained by the intranet administrators. If the user profile item doesn't exist, the replication code will create it.

You can insert the code, which changes a user profile, into the processor of the `intranetUpdateUserProfile` pipeline. This pipeline is run every time you create a new profile item. The newly created profile item and Sitecore user are then passed as parameters to the processor of the `intranetUpdateUserProfile` pipeline.

For more information about Active Directory integration, see the section [3.4, Extending the Portal](#)

You can extend SIP by integrating it with Active Directory. You can also extend SIP so that user profile fields are filled in automatically.

Active Directory Integration.

3.1.2 Manually Creating User Profiles

Administrators will need to manually create a user account from time to time.

To manually create a user profile:

1. In the **User Manager**, create a new user account. For most usage scenarios, simply specifying the username, password and assigning roles will be sufficient.
2. Create a *UserProfile* item under the *UserInfo* item. The item name must be the same as the name of the user account.

The *UserProfile* item is automatically moved to the correct location in the user profile hierarchy when the user logs in the next time.

3. Grant the user profile *Write* permissions to the newly created user. Otherwise, the user will not be able to edit their profile information.
4. Grant *Create* permission for the user profile if you want the user to be able to create a personal weblog.
5. Fill in the relevant fields of the user profile. Notice that the e-mail address, full name, and so on displayed on the intranet must be specified in the user profile item and in the user account in the **User Manager**.

3.2 Setting Security for User Profiles

Sitecore's standard security model applies to user profiles in the same way as to other items. This means that a user can only edit their user profile using the Page Editor if they have *Read* and *Write* permissions for the user profile item. It also means that you can grant certain employees (such as webmasters, managers, secretaries, and so on) permission to modify the user profiles of other employees.

We recommend that you only grant *Read*, *Write*, and *Create* permission for user profiles to intranet editors. We also recommend that you don't rename or delete user profiles.

When a user profile item is automatically created by the Intranet Portal (for instance, when a user logs into the intranet and no profile exists or when the information is replicated from Active Directory), the common permissions are normally granted, so that the user can edit their own profile and can create a personal weblog.

3.2.1 Default Security Settings

In the default installation, the **extranet\Everyone** role has the following permissions to the *UserInfo* item:

- Allow read
- Deny write, rename, create, delete

This means that every employee can view all the user profiles, for example when they view author information or search the phonebook, but the employees cannot modify the user profiles of other users.

3.2.2 Resetting Security Permissions for every User Profile

You will occasionally need to reset the security permissions on every user profile. To reset the permissions, navigate to the `/sitecore/admin/GrantUserProfilePermissions.aspx` page and click **Grant access**. Notice that you must be an administrator to run this script.

Important

Running this script grants every user read, write, and create permission to their own user profiles. Every other security setting that has been set on user profiles is removed. The script does not change any of the security permissions for the **UserInfo** item.

3.2.3 Main Sitecore Intranet Portal Roles

SIP contains a list of users' roles which make it easier to configure the system. Some of the roles are described in this section.

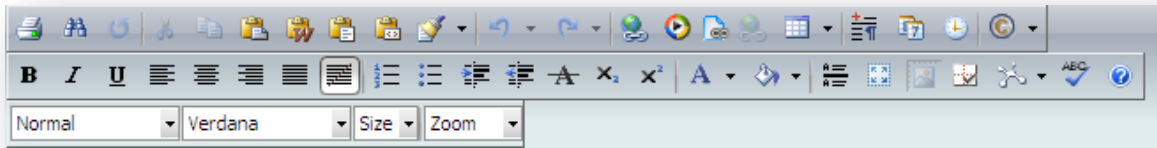
Sitecore\Sitecore Client Page Editor Only

This role grants users access to the **Page Editor** while limiting access to Sitecore Client (Desktop, Content Editor, and so on).

The *extranet\Employee* role has this role assigned by default. Extranet users who are assigned the *extranet\Employee* role can use the Page Editor to edit content items. .

Sitecore\Intranet Full Rich Text Editor

Members of this role have the access to all the features in the Rich Text Editor:



Sitecore\Intranet Minimal Rich Text Editor

Members of this role access to a limited set of features in the Rich Text Editor:



If a user is a member of both the *Sitecore\Intranet Full Rich Text Editor* and the *Sitecore\Intranet Minimal Rich Text Editor*, he only has limited access to the Rich Text Editor tools as specified in the *Sitecore\Intranet Minimal Rich Text Editor* role.


If the user is neither a member of the *Sitecore\Intranet Full Rich Text Editor* nor the *Sitecore\Intranet Minimal Rich Text Editor* role he only has access to the following tools of the Rich Text Editor :

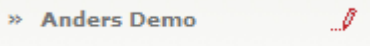



3.3 Editing User Profiles

This section describes how to allow employees to edit their user profiles and how to add fields to user profiles.

3.3.1 Allowing Employees to Edit their User Profiles

Users can edit most of the fields in their user profiles by clicking the **Edit Content**  button next to the user name.



>> Anders Demo 

Some of the user profile fields (such as name, phone number, e-mail address) are by default only editable in the **Content Editor** since these fields should only be updated by administrators. You can select which fields should be available in the Page Editor — select or clear the **Show this field in the Properties dialog** field for the fields of the `Intranet.UserProfile` template. To prevent users from editing certain fields in their user profile, set the appropriate rights on the template's fields.

By default, a user can edit the following fields when using the Page Editor

Users can edit the following fields by default:

Field Name	Description
Title	The user's job title
Date of Birth	The user's date of birth
Local phone number	The user's local phone number
Canteen	The user's favorite canteen. When the user navigates to the canteen item, this canteen is displayed automatically
Image	The picture of the user
Interests	The interests of the user
Preferred Language	The user's preferred language for displaying the content

A user clicks **Properties** to edit the following fields:

Field Name	Description
Department	The user's department
Location	The user's location
Phone	The user's phone number
Email	The user's email address

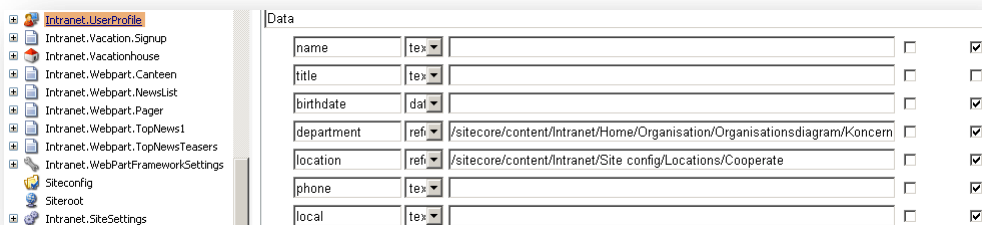
The **Job Description** Section:

Field Name	Description
Purpose	The purpose of this user's job
Responsibilities	The user's working responsibilities
Measure points	Measure points for the user
Key employees	The key employees this user is connected with
Refers to	A reference to the manager to whom this employee refers

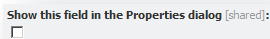
The **Data Channels** section is only available in the **Page Editor**, not when editing user profiles using the **Content Editor**. In this section a user can select their areas of interest.

3.3.2 Adding Fields to User Profiles

The user profile fields are stored in the `Intranet.UserProfile` template.



To allow a user to edit the new fields on the intranet, select the **Show this field in the Properties dialog** check box on the template:



Adding Fields to Intranet.Employee.xslt

After adding some new fields, you must modify the `Intranet.Employee.xslt` to show the new fields.

If you wish to localize the field labels, add the relevant fields to the `MainSettings` template (under the `dlheader` field) and use the new labels in the XSLT file.

Making Fields Searchable in the Phonebook and Pager

To make new user profile fields searchable and to have them displayed in the search results of the Pager or the Phonebook, you must customize SIP. For more info go to [Sitecore Developers Network](#) and see the Sitecore Intranet Content Features document.

3.4 Extending the Portal

You can extend SIP by integrating it with Active Directory. You can also extend SIP so that user profile fields are filled in automatically.

3.4.1 Active Directory Integration

For information about the recommended ways of integrating Sitecore Intranet Portal with the Active Directory, please refer to the [Active Directory Integration Guide on the Sitecore Developers Network](#).

Sitecore Intranet Portal provides the sample code for Active Directory integration. To download this code, go to [Sitecore Developers Network](#).

3.4.2 Setting Location, Department and Canteen Automatically

Some of the user profile fields (location, department, and canteen) can be filled in automatically from external sources such as Active Directory.

An example of the code which sets a canteen based on Active Directory roles can be found in the sample [Active Directory integration code on the SDN](#).

Chapter 4

Skinning

You use skins to change the look and feel of the Sitecore Intranet Portal. Skins do not change any of the functionality in the product or the content itself.

For example, along with the default skin, the portal could have a skin for the Christmas holiday or a skin for a new product launch.

The skinning functionality makes it easy to create skins and change them. This chapter describes how to configure the skinning engine and extend its default functionality for more complex requirements.

A skin consists of a number of images, texts paragraphs, and CSS styles.

This chapter contains the following sections:

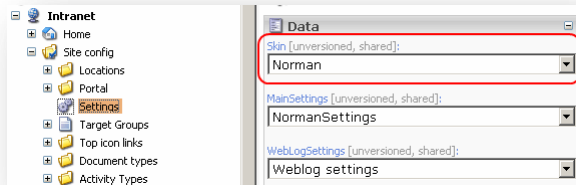
- Skins
- Customizing Style Sheets
- Developer Hints

4.1 Skins

Sitecore Intranet Portal lets you change the skin that the whole product uses as well as assign individual skins to different departments and locations. In this section you can find information about customizing the skin.

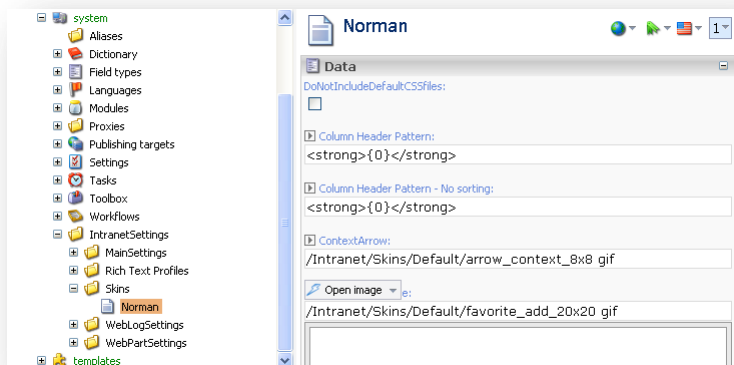
4.1.1 Changing Skins

Administrators use the **Content Editor** to set skins. The Skin setting is located in the *Settings* item under the `/Intranet/Site config` item:



All an administrator needs to do is to select the skin from the drop-down list and the site design of the intranet site is changed after the next time you refresh the browser.

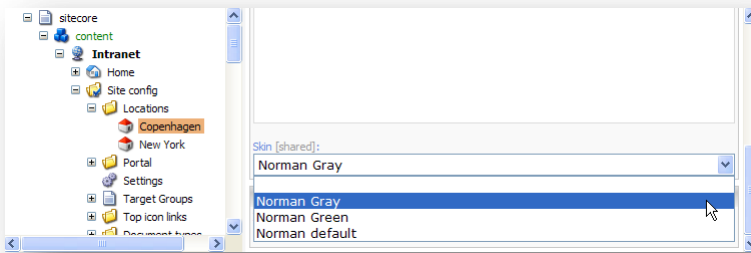
Skins are stored under the `/system/IntranetSettings/Skins/` item. Each skin contains a set of design elements including icons, site design images, header patterns, button titles, login page design, and stylesheets.



Every element in a skin can look different in different languages — this is extremely useful for elements such as buttons with titles.

4.1.2 Department and Location Based Skins

Skins can be assigned to departments and locations. A user belonging to a department or a location that has a skin associated with it will automatically see that skin.

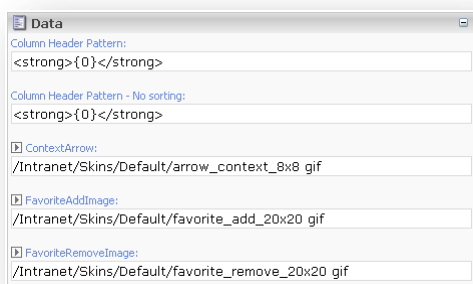


By default, department skin takes precedence over location skin.

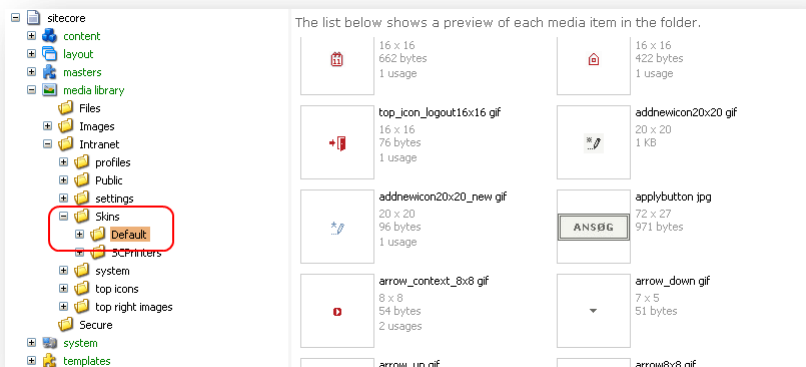
Skin resolving is controlled by the `intranetSkinResolver` pipeline in the `web.config` file. Developers can change the order of the department and location resolvers, or they can implement custom skin resolvers to assign skins based on other parameters.

4.1.3 Customizing Icons and Images

Icons and images are also defined as part of the skin. For example, in the following screenshot, you can see that some images have been defined for this skin:



While the images can be stored anywhere in the media library, we recommend that you create one icons folder per skin under the `/media library/Intranet/Skins` folder.



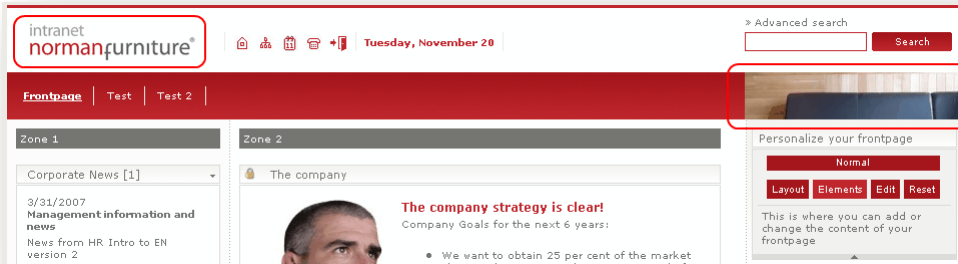
The following images that are typically customized in a skin:

- PrintIcon

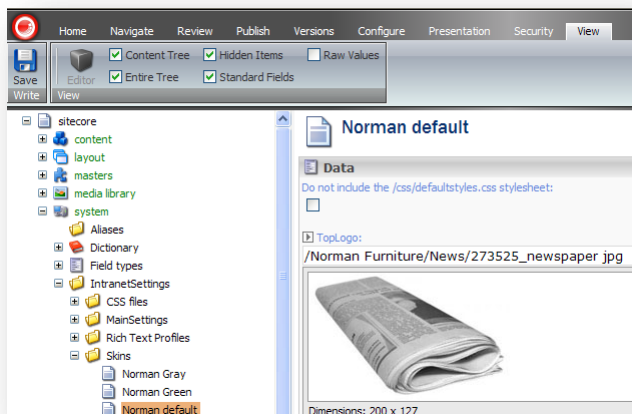
- TopSearchbuttonImage
- AddNewItemIcon
- FrontpageLinkImage
- LoginBackgroundImage

4.1.4 Changing the Logo

In most skinning scenarios the customer wants to change the logo and the picture in the top right-hand corner to reflect the company's identity:



The logo is defined by the **Top logo** field in the skin item:



In the Norman Furniture example, the size of the logo is 217x40. To use a different size for the logo, you must change the `.TopLogoCell` style as well, because this style defines the height of the image.

Note

For backwards compatibility, the logo can also be defined by the *Top logo* field of a *MainSettings* item. This value is used if the *Top logo* field on the current skin is empty.

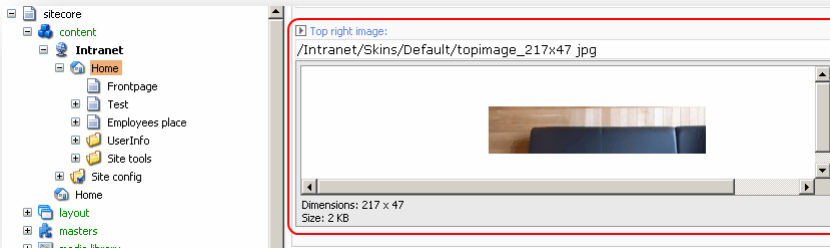
4.1.5 Changing the Top Right-hand Picture

You can use different pictures in the top right-hand corner to differentiate the different sections of the intranet portal:



Every item that is based on the `Intranet.Section` template (and the `Intranet.Frontpage` template) has a `Top right image` field where you define the image for that particular section. If no image is set on the section, the image from the front page item is used.

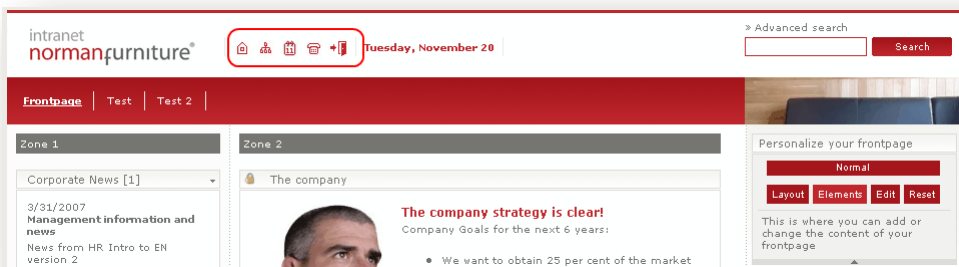
The default top right-hand image should therefore be set on the root item (`/content/Intranet/Home`) in the `Top right image` field.



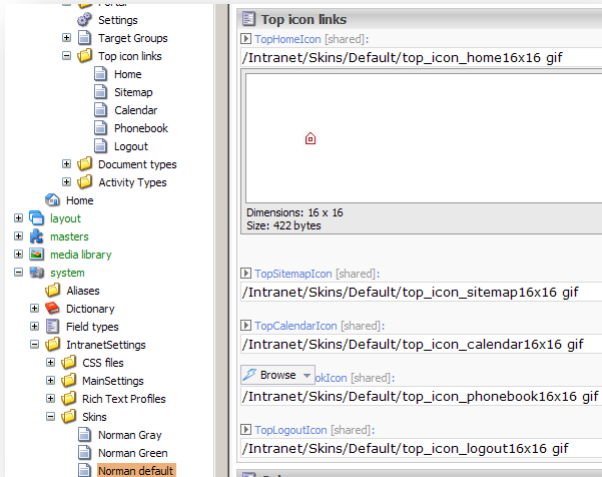
Note
The top right-hand picture is hardcoded to size 217x47 by default. To use a different size, modify the section where the image tag is defined in the `/xsl/Intranet.TopMenu.xslt` file.

4.1.6 Customizing the Icons at the Top of the Page

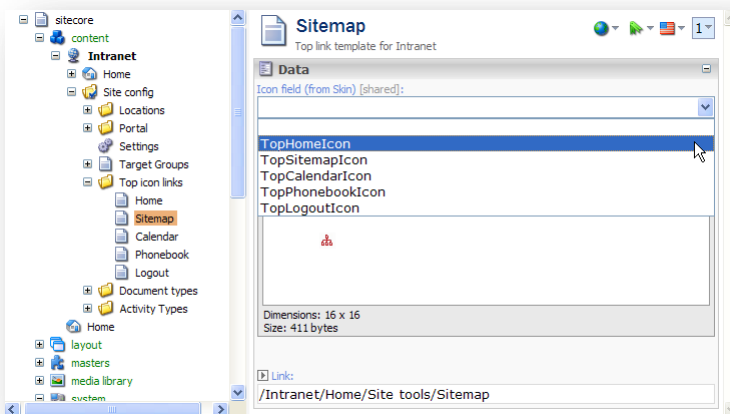
The icons displayed at the top of the page are controlled by the current skin:



The images are defined in the skin item:



The icons displayed at the top of the page are called *Top icon links* and you select the image that should be displayed in the **Icon** field of each *Top icon link* item.



For backwards compatibility, you can still select an image in the **Icon** field. This value will be used if the *Icon field (from Skin)* field is empty.

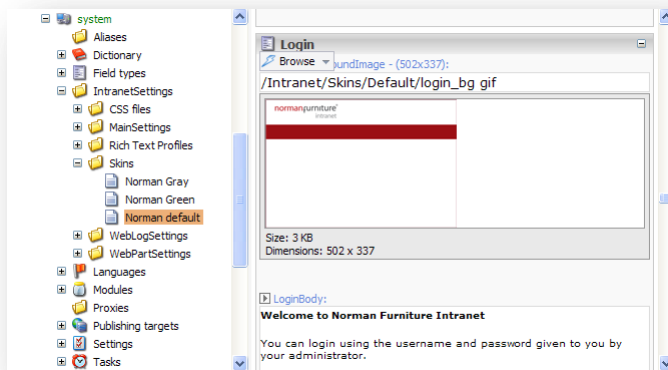
The `Intranet.TopLink` template is used for top icon links.

Note
The **ALT** text for the top icons is not used. Instead, the **Title** of the link field used as the **ALT** text.

4.1.7 Customizing the Login Page

The login page that is used for each skin is defined in the skin item. The skin items are stored in the `System\IntranetSettings\Skins` folder. Typically you want to change the text and the logo. The logo is a part of the *LoginBackgroundImage*, so you have to create a custom background image

with your company logo. The Login page text can be modified in the **LoginBody** field:



The *Login* fields on the skin are only used by the default skin in the default language for the Web site (typically the language selected in the **Corporate Language**). These fields are not used for all other skins and languages.

4.1.8 Creating a Custom Skin

There are two ways to create a custom skin.

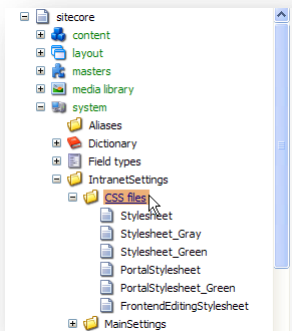
You can:

- Create a new item based on the `Intranet.Skin` template and fill out all the fields in the item. This can be quite time-consuming.
- Duplicate the existing default skin and only change what needs to be modified.

4.2 Customizing Style Sheets

An important element of skins is cascading style sheets (CSS). The style sheets allow you to tweak the design of the portal in greater detail.

By default, the CSS definitions are stored as physical files. The available CSS files are defined in the `/sitecore/system/IntranetSettings/CSS files` folder.



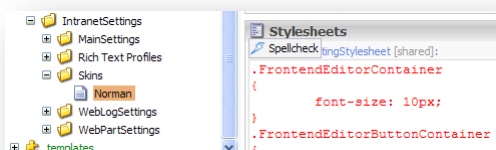
On the skin item, you configure the CSS files to include in the **IncludedCSSFiles** field in the **Stylesheets** section.

Each file is pre-processed before it is sent to the client. By default, Sitecore Intranet Portal changes `~/skinimage/` by adding a skin GUID to avoid redundant requests. This is done by the `parseStylesheet` pipeline. Developers can implement custom processors that will be run during preprocessing of CSS files. This lets developers assign special settings to display intranet portal pages for specific users. For example, to set special font size for a specific user, based on their profile preferences.

For backwards compatibility, three text fields are available on the skin item that corresponds to each of the standard CSS files. These fields are empty by default, but if you add CSS definitions in them, these definitions are included on the page. This can be used for selecting the CSS definitions that administrators want to edit with the **Content Editor**.

This tutorial does not describe the concept behind stylesheets, but rather how stylesheets are used in the skinning engine.

The stylesheet definitions are stored in the skin item in the **Stylesheets** section.



You can change there style sheets:

- FrontendEditingStylesheet
- PortalStylesheet
- Stylesheet

Each of these stylesheets is described in detail in the following sections.

The stylesheets are dynamically added to the corresponding layouts by the `/layouts/Intranet.Skin.ascx` control.

FrontendEditingStylesheet

This style sheet only applies to SIP 2.2. For more information about this style sheet, see the [SIP 2.2 documentation](#).

PortalStylesheet

The portal stylesheet is included in the portal front page and determines the look and feel of Web Parts as well as the Web Part framework.

```

PortalStylesheet [shared]:
/* ----- */
/* - WebPart: DynamicContentTeaser - */
/* ----- */

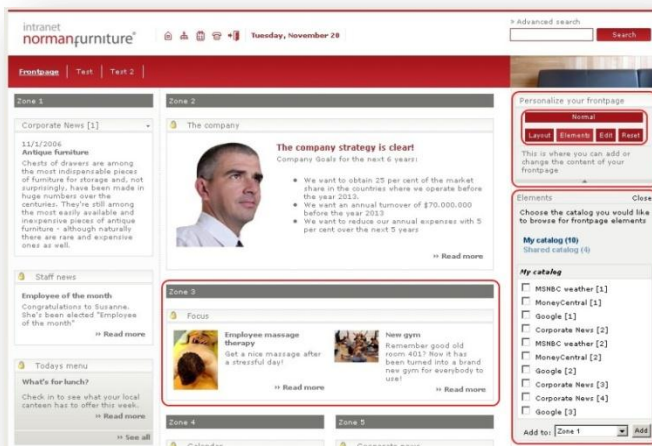
.DynamicContentTeaserContainer
{
    padding: 6px 6px 6px 6px;
    width: 100%;
}
.DynamicContentTeaserImageCell
{
    padding-right: 6px;
    width: 65px;
}
.DynamicContentTeaserImageCell IMG
{
    width: 65px;
}

```

The stylesheet consists of the following areas:

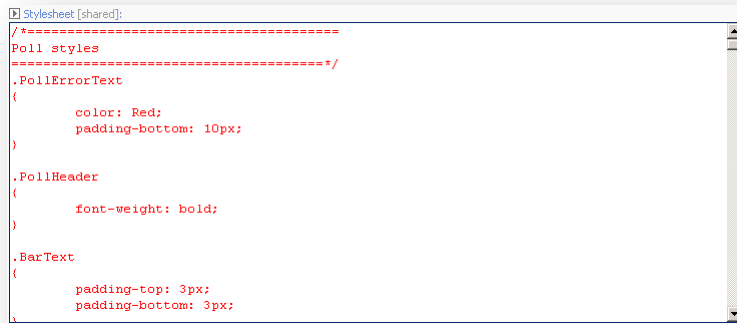
- WebPart: DynamicContentTeaser
- WebPart: DynamicContentRotator
- WebPart: DynamicContentList
- Portal Toolbox
- Portal Framework
- Portlet DropZone
- Portlet Empty
- Catalog Zone
- Mode buttons
- Editor Zone

Some of the elements that are controlled by this stylesheet are highlighted in the following image:



Stylesheet

This is the base stylesheet that defines the overall look and feel of the portal.



```
Stylesheet [shared]:
/*=====
Poll styles
=====*/
.PollErrorText
{
    color: Red;
    padding-bottom: 10px;
}

.PollHeader
{
    font-weight: bold;
}

.BarText
{
    padding-top: 3px;
    padding-bottom: 3px;
}
```

The style sheet consists of the following areas:

- Poll
- FAQ
- SendPage (Tip a friend)
- Login
- Weblog
- TopSearch
- TopMenu
- Top
- Newslist
- NewsFrontpage
- Forum
- Employee
- ContentPageList
- Canteen
- Author
- Activities
- VacationSignUp
- VacationHouse
- UpdateList
- MarketPlace
- Print
- PhoneBook
- PersonalContext
- NewsArchive
- EmployeeList
- DownloadList

- DocumentList
- Breadcrumbs
- Slideshow (search for ".Thumbnail")
- Sitemap
- Submenu
- Text styles (general styles)

The styles are categorized by their functionality.

You would typically modify the following styles (in the *Stylesheet* field):

- The styles used in the header of the page
 - .TopDateCell (1/4 down the stylesheet)
 - .AdvancedSearchLink (3/5 down the stylesheet)
 - .TopSearchField (1/4 down the stylesheet)
- Submenu arrow
 - ul#submenumain li submenulinkon (4/5 down the stylesheet)
 - ul#submenumain li submenulinkoff (4/5 down the stylesheet)
- Horizontal line at the top and the bottom
 - .MainBottomCell (2/3 down the stylesheet)
 - .maintable (2/3 down the stylesheet)
- Top menu background
 - .maintabletopmenucell (2/3 down the stylesheet)

Also, you would typically adjust the text colors (in the **Stylesheet** field):

- .maintabletopcell (2/3 down the stylesheet)
- Login box
 - .LoginBodyCell
 - .LoginText
 - .LoginTextSmall
 - .LoginField (often darker since it is the color of text in input fields)
- The headline of TopNews1 Web part is defined in the **PortalStylesheet** field
 - DIV.WebPartContentTopNews1Header

4.2.1 Referencing Skin Images from CSS Files

To make dynamic references from CSS files to skin images, use the following syntax:

```
~/~/skinimage/[fieldname].ashx.
```

This returns the correct image based on the user's current skin.

4.3 Developer Hints

Here are some tips that should be useful for developers who are working with SIP.

4.3.1 Dynamic Image Scaling

Dynamic image scaling helps to keep the layout consistent and reduce bandwidth usage, even if the user uploads/selects large images:

- The default maximum width is specified in the **MaxImageWidth** field in the **MediaLibrary** section of the main settings (default: 300 pixels)
- Maximum width of list images are configured in the **ListMaxImageWidth** field in the **MediaLibrary** section of the main settings (default: 65 pixels)
- Maximum width of blogger image is configured in the **bloggerimagemaxwidth** field on the **WebLog Settings** item (default: 250 pixels)
- Maximum width of profile images is 200 pixels (`Intranet.Author.xslt`)
- A `<sc:scaledImage field="[fieldname]" />` XSL control is available for easy image scaling. It takes the same parameters as the `<sc:image>` control, but automatically uses the default `MaxImageWidth` setting.

Note

You can allow editors to specify the maximum width on an image-by-image basis by creating a field called `[fieldname]Width`. The `<sc:scaledImage>` control will automatically use this value if specified.

4.3.2 Using Skinned Images in XSLT Files

When you use skinned images in XSLT files, you should remember that:

- A new `<sc:skinnedImage>` XSLT control makes it easy to render skinned images.
- The syntax is `<sc:skinnedImage field="[fieldname]" useFallback="[true|false]" style="[style definition]">`.
- The following parameters are used:
 - `field` (name of image field on skin item)
 - `useFallback` (will read field from main settings item if field is empty on skin)
 - `style` (for example `style="vertical-align: middle;"`)
- If you specify `useFallback` and the field is empty on the skin, the field will be read from the `MainSettings` item instead (see `Intranet.Top.xslt` for an example).

Chapter 5

Web Parts

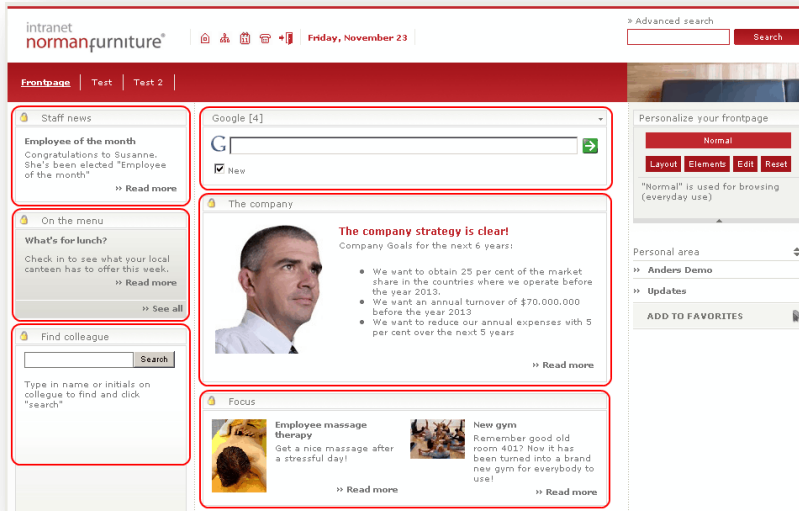
You can customize the intranet portal using various mechanisms. This chapter describes how to modify your site with Web Parts.

This chapter contains the following sections:

- Web Parts Configuration
- Standard Web Parts
- Web Parts, Sublayouts and Web Controls
- Extending the Portal. Creating Your Own Web Parts
- Extending the Portal. Using SharePoint Web Parts

5.1 Web Parts Configuration

Web Parts are the portal building blocks. A Web Part represents a block of functionality. They are used to create portals and dashboards. The Web Part framework is one of the most important aspects in Sitecore Intranet Portal. Take a look at the screenshot showing a portal with various Web Parts:



Web Parts are easy to use and reuse because they are self-contained.

Depending on the implementation and settings, end users can customize their own portal by designing their own view and content. In the screenshot above this can be seen through the controls under the headline **Personalize your frontpage**.

The Web part implementation in Sitecore Intranet Portal is built on top of the standard asp.net 3.5 Web Parts framework with the following extensions:

- Standard Sitecore renderings can be used as Web Parts
- Web Parts are configured using Sitecore items
- Sitecore security model is used to restrict access to the Web Parts
- User specific information and settings can be accessed easily using the Sitecore API or XslHelper functions.

Web Parts should be built using one of the following templates:

- **Web Part**
Use for "normal" Web Parts that are located in a given assembly as code.
- **Web Part Node**
Use for wrapping a Web Part in a configuration so that a generic Web Part can be used in multiple scenarios.
- **ReferenceRendering**
Use to use standard renderings such as XslRendering, XmlControl, UrlRendering and MethodRendering as Web Parts.

The following paths are important when working with Web Parts in Sitecore:

Path	Description
<i>/templates/WebPart Framework</i>	Here are all the Web Part specific templates located.
<i>/templates/Intranet</i>	Here are all the templates for the Intranet is located. This also contains a lot of predefined templates for Web Part Nodes (look for <code>Intranet.Webpart</code>)
<i>/system/Modules/Web Parts</i>	Here are all the installed Web Parts located. The employees can add these Web parts to their personal portal by clicking "Elements > Shared catalog".
<i>/content/Intranet/Site config/Portal</i>	The Web parts in this location define the default layout of the portal, including Web parts that the users are not allowed to remove.

5.1.1 Security and Rights Management

The WebPart framework uses the Sitecore CMS security model. Access rights can be applied to both Web Parts themselves and to Web Part configurations.

Web Parts located under the `/content/Intranet/Site config/Portal` item are visible on the front page by default. Web parts located under the `/system/Modules/Web Parts` item can be added to the front page by users.

If a user has the write permission on a Web part, then such user can move the Web Part to a different location on the front page, or one can close the Web part so that it is no longer visible.

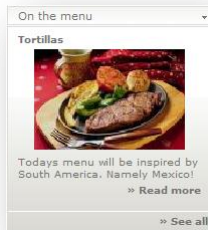
A user should have **Read** permissions on a Web part to be able to see it.

5.2 Standard Web Parts

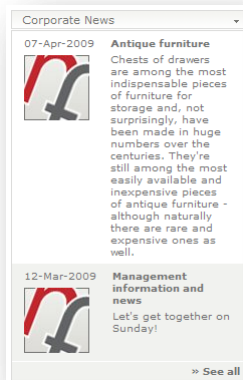
Sitecore Intranet Portal comes with a number of Web Parts out of the box.

Every Web part is built on top of a template and a sublayout. The following list shows the templates that are used for building Web parts. Every template inherits from the `Web Part Node` template. Corresponding sublayouts are named the same way as the templates:

- `Intranet.Webpart.Canteen`

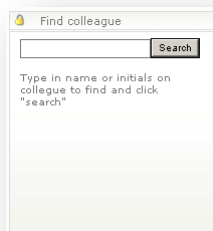


- `Intranet.Webpart.Newslist`



Notice that this Web part will only display news items that have a date filled out.

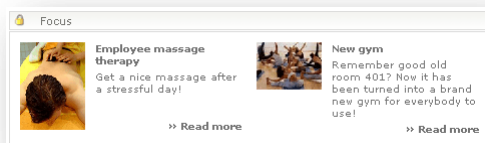
- `Intranet.Webpart.Pager`



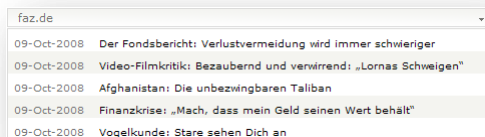
- **Intranet.Webpart.TopNews1**



- **Intranet.Webpart.TopNewsTeasers**



- **RSSWebPart**



There is also the `WebPart` template that has some branch templates defined. The template consists of a type reference to a programmatic Web Part (signature of the class containing full namespace and assembly). The branch templates defined are:

- **Intranet.WebPart.Content List**
Shows a list of content
- **Intranet.WebPart.Content Teaser**
Shows an abstract of some content
- **Intranet.WebPart.Content Rotator**
Toggles between different content items
- **Intranet.WebPart.DWP Wrapper**
Wrapper for SharePoint type Web Parts. Currently only SharePoint 2001 Web Parts are supported

5.2.1 RSS Web Part

The RSS Web part allows administrators to make pre-configured RSS feeds available to the users of the intranet. With all basic elements being already pre-configured for this standard Web part, you can create a RSS feed with unprecedented ease.

Two types of RSS Web parts can be configured:

- Usual RSS feeds
- Customizable RSS feeds

5.3 Web Parts, Sublayouts and Web Controls

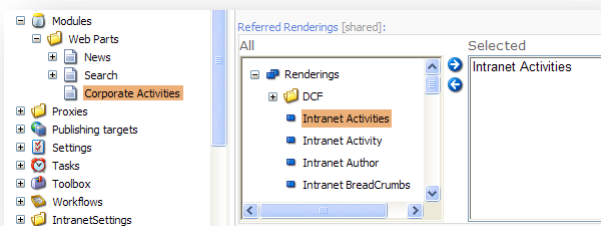
There are two different ways to work with Web Parts:

- Create "real" Web Parts that are built on the `WebPart` template.
- Use existing renderings as Web parts using the `ReferenceRendering` template.

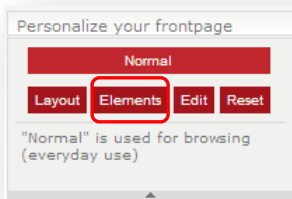
The later approach is described in the following section.

5.3.1 Example: Using a sublayout as a Web part

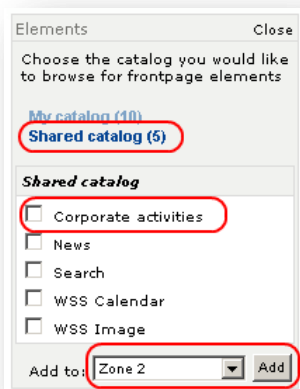
Let's create a new Web Part that shows activities from the calendar. We will start by creating a new item called *Corporate activities* based on the `ReferenceRendering` template under the `/system/Modules/Web Parts` item. The `ReferenceRendering` template is located under the `/templates/WebPart Framework/` item. Select the `Intranet Activities` rendering and save the item.



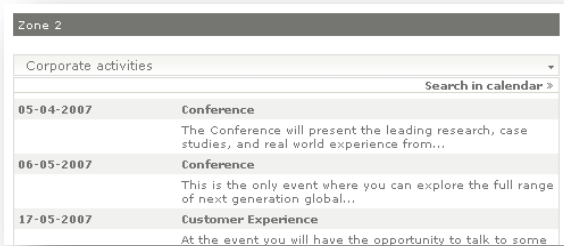
Go to the portal front page and click **Elements** on the **Personalize your frontpage** Web part:



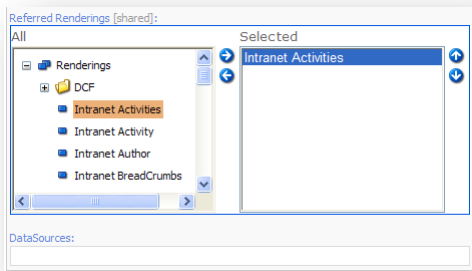
Click **Shared catalog** and then click **Corporate activities** that we have just created. Select a zone (zone 2 in our example) and click **Add**.



Now a Web Part which includes the rendering is inserted into the portal front page.

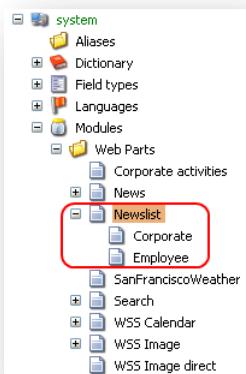


Notice that the `Intranet Activities` rendering already has a default data source. With other renderings it might be necessary to set the `DataSources` field of the item built on the `ReferenceRendering` template:



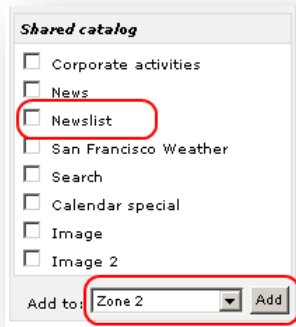
5.3.2 Example: Creating Web Parts Based on the Web Part Template

To create a new Web part, create a new item based on the `Intranet.WebPart.Content List` template under the `/system/Modules/Web Parts/` item. This creates a new Web Part designed for the news lists. Call the item `Newslist`. In this example we will let a users decide between two different lists trough simple configuration, so the next step is to create the available configurations. Create two items below the `Newslist` item both based on the `Intranet.WebPart.Content List Config` template. Call one of them `Corporate` and the other `Employee`.



Each configuration has a `List source` field. Select a location where a list of news is stored. In this example the news are created under the `sitecore/content/Intranet/Home/Test/Corporate` news item and the `sitecore/content/Intranet/Home/Test/Employee` news item. Also give the configurations some proper headlines.

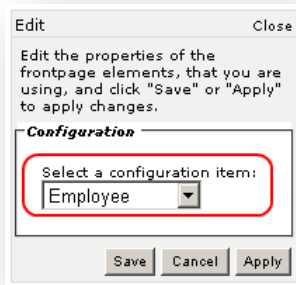
Go to the portal front page and click **Elements** to add the new Web Part.



After the new Web Part has been added it should pop up in the portal and look something like the screenshot below:



Click **Edit** to edit the page, and then click the **Web Part**. Now the user has an option to select either the corporate or the employee news.



5.4 Extending the Portal. Creating Your Own Web Parts

Since the Sitecore Web Part framework is based on the standard ASP.NET 2.0 Web Part framework, the techniques used in the creation of the standard Web Parts can be used in Sitecore Intranet Portal as well. While Sitecore accepts standard Web Parts inheriting from `System.Web.UI.WebControls.WebParts.WebPart` it also provides its own base class `Sitecore.Modules.WebPartFramework.Web.WebParts.WebPart` that provides access to Sitecore specific methods. A Sitecore Web Part overrides the `CreateEditorParts` method and adds a couple of properties. The override and the properties are used in order to get the special Web Part configuration running that enables Web Parts to use Sitecore items as configuration. The full list of properties in the Web Part class is listed in the following table:

Property	Description
<code>ConfigItem</code> (get only)	Refers to the current selected configuration item. All available configurations are shown in a dropdown list when editing the Web Part (courtesy of the method <code>CreateEditorParts</code>).
<code>ConfigItemID</code> (get and set)	The id of the <code>ConfigItem</code>
<code>ReferredItem</code> (get only)	Refers to the <i>Web part node</i> item. The Web part node is typically placed in the portal configuration defining the default Web parts. This property is rarely used.
<code>ReferredItemID</code> (get and set)	The id of the <code>ReferredItem</code>
<code>WebPartItem</code> (get only)	Refers to the <i>Web Part</i> item itself. It is the item that builds on the <code>WebPart</code> template.
<code>WebPartItemID</code> (get and set)	The id of the <code>WebPartItem</code> .

The following code implements a simple Web Part. The Web Part simply retrieves a 5 day weather forecast for the San Francisco area. As a next step, you could implement caching and custom configuration. Notice that the `ndfdXML` class is generated by the `wsdl.exe` from the URL: <http://www.weather.gov/forecasts/xml/DWMLgen/wsdl/ndfdXML.wsdl>

Note that this simple Web Part does not use Sitecore specific features.

```

/// <summary>
/// Simple Sitecore Web Part that fetches a five day weather
/// forecast for the San Francisco area.
/// A more simple Sitecore adapted version of:
/// http://aspnet.4guysfromrolla.com/articles/030205-1.aspx
/// </summary>
public class WeatherWebPart : WebPart
{
    private decimal SAN_FRANCISCO_LATITUDE = 37.766667M;
    private decimal SAN_FRANCISCO_LONGITUDE = -122.433333M;

    private string serviceUrl =
"http://www.weather.gov/forecasts/xml/SOAP_server/ndfdXMLserver.php";
    private ndfdXML _service = new ndfdXML();

    private int days = 5;
    private DateTime date = DateTime.Now;
    private Literal literal;

    /// <summary>
    /// Creates the controls needed for the WebPart to function
    /// </summary>
    protected override void CreateChildControls()
    {
        _literal = new Literal();
    }
}

```

```

        Controls.Add(_literal);
    }

    protected override void Render(System.Web.UI.HtmlTextWriter writer)
    {
        base.Render(writer);
    }

    /// <summary>
    /// Fills the data into the control
    /// </summary>
    /// <param name="e"></param>
    protected override void OnInit(EventArgs e)
    {
        base.OnInit(e);
        EnsureChildControls();
        string data = GetData();
        // simple error checking...
        // only format the result if there hasn't occurred an exception
        if (!Regex.IsMatch(data, "exception", RegexOptions.IgnoreCase))
        {
            data = FormatResult(data);
        }
        //TODO: Build the controls here...
        literal.Text = data;
    }

    /// <summary>
    /// Returns xml data from the weather webservice
    /// </summary>
    /// <returns></returns>
    private string GetData()
    {
        //TODO: Add caching...
        service.Url = serviceUrl;
        string result;
        try
        {
            result = _service.NDFDgenByDay(SAN_FRANCISCO_LATITUDE, SAN_FRANCISCO_LONGITUDE,
date, days.ToString(), formatType.Item24hourly);
        }
        catch (Exception ex)
        {
            result = ex.ToString();
        }
        return result;
    }
    /// <summary>
    /// Formats the xml response
    /// </summary>
    /// <param name="str"></param>
    /// <returns></returns>
    private string FormatResult(string str)
    {
        // load up the XML data...
        XmlDocument doc = new XmlDocument();
        doc.LoadXml(str);

        // read in the data
        XmlNodeList highs =
doc.SelectNodes("/dwml/data/parameters/temperature[@type='maximum']/value/text()");
        XmlNodeList lows =
doc.SelectNodes("/dwml/data/parameters/temperature[@type='minimum']/value/text()");
        XmlNodeList cloudIcon = doc.SelectNodes("/dwml/data/parameters/conditions-
icon/icon-link/text()");

        StringBuilder sb = new StringBuilder();
        string cloud;
        int high;
        int low;
        sb.Append("<table><tr>");
        for (int i = 0; i < _days; i++)
        {

```



```
cloud = cloudIcon[i] != null ? cloudIcon[i].Value : string.Empty;
high = highs[i] != null ? FahrenheitToCelsius(int.Parse(highs[i].Value)) : 0;
low = lows[i] != null ? FahrenheitToCelsius(int.Parse(lows[i].Value)) : 0;
sb.AppendFormat("<td>{0}<br/><img src=\"{1}\"/><br/>{2}<br/>{3}</td>",
_date.AddDays(i).ToShortDateString(), cloud, high, low);
}
sb.Append("</tr></table>");
return sb.ToString();
}

/// <summary>
/// A simple converter from Fahrenheit to Celsius since the result uses
/// Fahrenheit degrees.
/// </summary>
/// <param name="fahrenheit"></param>
/// <returns></returns>
private int FahrenheitToCelsius(int fahrenheit)
{
    return (int)(((double)(fahrenheit + 40) * 0.5) * 1.1) - 40;
}
}
```

5.5 Extending the Portal. Using SharePoint Web Parts

Since Sitecore is based on the standard ASP.NET 2.0 Web Part framework, Web Parts from other sources can also be used. In general ASP.NET 2.0 Web Parts can be used since the Sitecore `WebPart` class actually inherits the standard `WebPart` class and therefore degrades transparently.

This means that Sitecore can use Web Parts developed for Microsoft SharePoint 2007 platform. However there are the following issues to be aware of:

- If the Web Part uses the SharePoint object model, the Sitecore server must reside on the same server as SharePoint since the SharePoint object model can only access the localhost server. In general, such a setup cannot be recommended due to scalability and performance issues.
- It is a common practice to sign Web Parts for SharePoint with a strong name. This is however not the preferred way in Sitecore CMS that is not signed. So if a Web Part should be used in both places, SharePoint should be configured to run in the "Full trust" mode.
- When using SharePoint version 2007 (Microsoft Office SharePoint Server 2007 - MOSS 2007 and Windows SharePoint Services 3.0 - WSS 3.0) it is recommended to use the ASP.NET 2.0 `WebPart` class as the base class for development because this ensures compatibility with Sitecore CMS. The previous versions of SharePoint used their own Web Part framework and therefore their own WebPart base class. This class IS STILL present in SharePoint 2007 but is only there for backwards compatibility. While Microsoft now recommends using the generic `WebPart` class when developing Web Parts, there is still a lot of available Web Parts that use the old base class. If these Web Parts are to be used, a wrapper along the lines of the `DWPWrapper` should be developed.
- At the time of writing, using a Web Part built on the Sitecore `WebPart` base class will fail when the Web Part enters the Edit mode in SharePoint. It is therefore recommended to build Web Parts based on the ASP.NET 2.0 generic `WebPart` class if compatibility is an issue, and the code does not need access to Sitecore specific configuration or items.

Chapter 6

Search

This chapter describes various techniques related to search implementation in Sitecore Intranet Portal.

This chapter contains the following sections:

- Configuring Indexing for PDF Files and Office Documents
- Customizing the Way Search Results are Displayed

6.1 Configuring Indexing for PDF Files and Office Documents

In order to enable full text indexing of Microsoft Office documents, the appropriate IFilter needs to be installed. This IFilter cannot be downloaded as a separate element; it is a part of the Microsoft Office package. Installing a client application on a server may however not be the best practice and there is also the question of the MS Office license. Another way is to install the Windows Desktop Search on the server (visit [Microsoft's Web site](#))

For indexing PDF files, you do not need to install an IFilter. The open source PDFBox is already installed as a part of Sitecore Intranet and works without any additional configuration.

6.2 Customizing the Way Search Results are Displayed

It is possible to customize the display of search results.

This can be done with a custom class that inherits from the `IntranetSearch` class and overriding the relevant methods such as `GetSearchResultsFormatter`, `GetSearchQuery`, `GetQueryString`.

Chapter 7

Editing Using the Page Editor

Sitecore Intranet Portal allows editors to edit content directly in the **Page Editor**. The process is designed to be easy, hiding the more complex challenges of content management.

Content administrators must use the standard Sitecore interface for more complex administrative scenarios such as moving or copying items around in the hierarchy and so on.

This chapter contains the following sections:

- Creating and Editing Content
- Managing Permissions for Page Editor Features

7.1 Creating and Editing Content

You can edit the content of the intranet site in the **Page Editor**. If you wish to use the old SIP editor instead of the **Page Editor**, set the `Intranet.UseOldFrontendEditor` parameter to true in the `web.config` file. For more information about editing the content using the old SIP editor, see the [SIP Core Concepts document for SIP 2.2 on the SDN](#).

In general, there are four main actions that users with the appropriate permissions can perform on items:

- Read — view the content
- Write — edit the content
- Create — create new content items
- Delete — delete content items directly from the **Page Editor**

You create content in the **Page Editor** with the help of branch templates assigned to the item. Branch templates allow you to create an item as well as a set of items.

If an item is editable and has branch templates assigned, a user with **Write** and **Create** permissions can see two buttons in the upper right-hand corner of the main frame:



These buttons are:

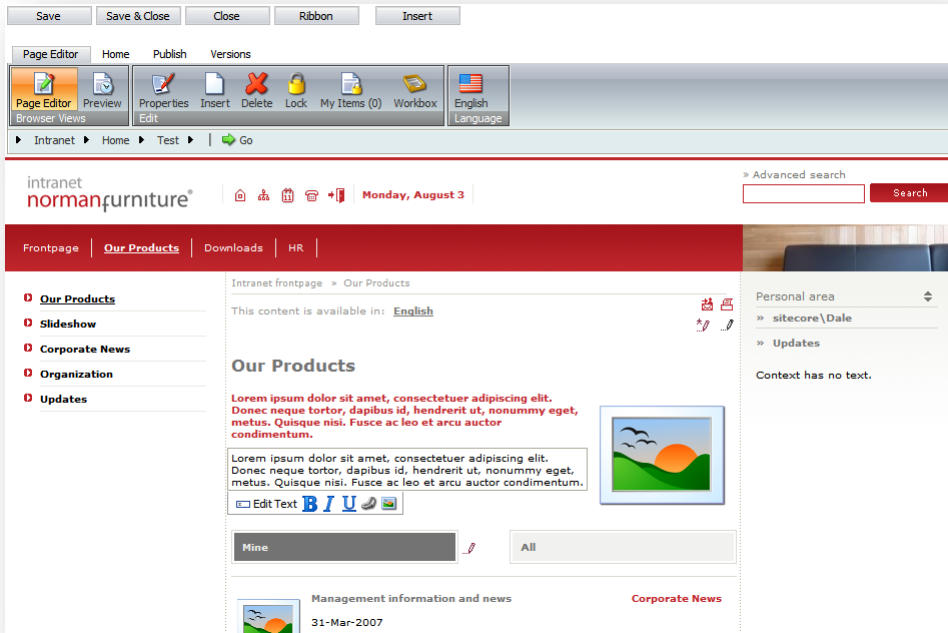
Create 

You use this button to create new content items under the current item (the new items become the children of the current one). When you click the **Create** button a pop-up window opens which allows you to select one of the branch templates available from this item and enter a name for the new item.

Edit 

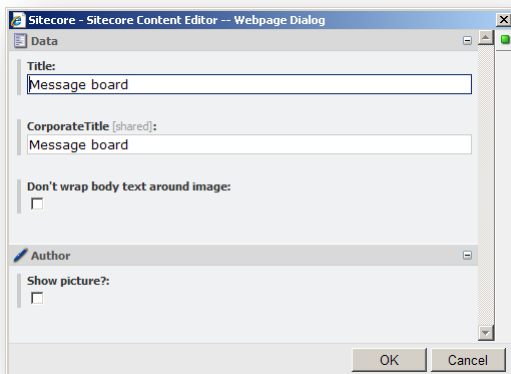
You use this button to edit the currently item.

Click **Edit** to activate the **Page Editor**:

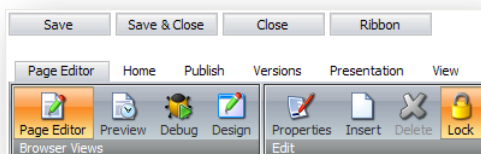


7.1.1 Editing Additional Fields

Inline editing allows you to change most of the fields that are visible on the page. Fields that are rarely changed and fields that are not visible on the page can be edited in the **Properties** dialog box. Examples of such fields are **Title**, **CorporateTitle**, **Don't wrap body text around image**, **Show author image**, **Weblog settings**, and so on



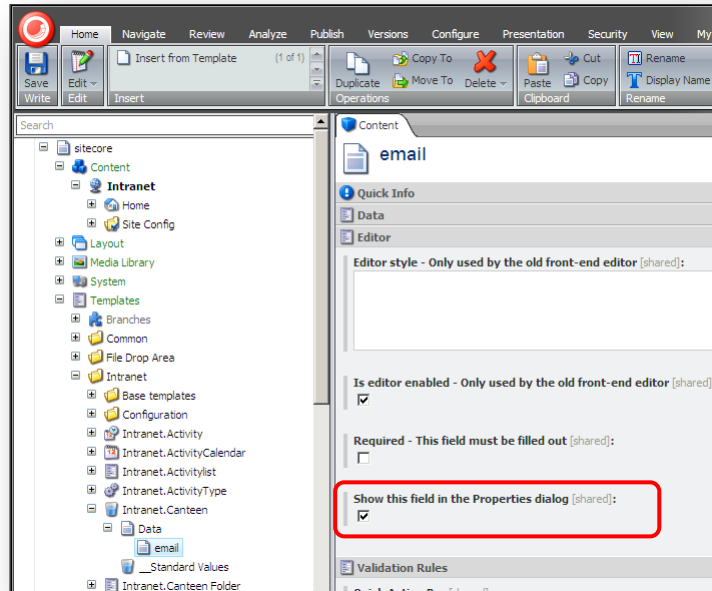
To open the **Properties** dialog box, in the **Page Editor**, in the **Edit** group, click **Properties**:



Intranet administrators and developers can add fields to the **Properties** dialog.

To add fields to the **Properties** dialog box:

1. In the **Content Editor**, navigate to the template that the item is based on.
2. Select the item of the field that you want to be displayed in the **Properties** dialog box.
3. In the **Editor** section, select the **Show this field in the Properties dialog** check box.




A template might not contain its own fields, but only inherit them, for example, the `Intranet.Content` template. In this case:


1. In the **Content Editor**, select the template item.
2. Click the **Inheritance** tab
3. Select the field that is inherited.
4. In the **Editor** section, select the **Show this field in the Properties dialog** check box.

7.2 Managing Permissions for Page Editor Features

You can use the Sitecore security system to restrict the access to the different **Page Editor** features.

The **Create**  button is only available when the following conditions are met:

- The user has **Create** permission for the current item
- One or more templates are assigned to the item
- The user has **Read** permission for at least one of the assigned branch templates

The **Edit**  button is only available when the user has **Write** permission for the current item.

The individual fields in the item template can only be edited in the **Page Editor** when the following conditions are met:

- The user has **field:write** permission for the field
- The field type can be edited in the **Page Editor**.

Chapter 8

Multilanguage Features

Sitecore Intranet Portal provides you with multilanguage features that allow you to make an international portal.

This chapter contains the following sections:

- Layered Translations
- Content Area vs. Menu Area
- Working with Different Languages
- Working with the Localized Placeholder
- Adding and Removing Languages
- Security and Languages

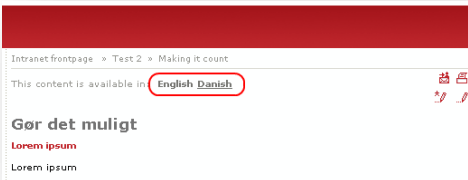
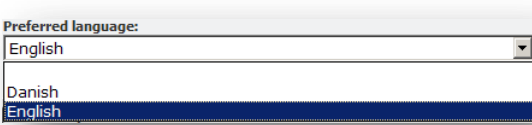
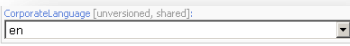
8.1 Layered Translations

For many international organizations, providing full featured local-language versions is a key challenge.

Sitecore Intranet Portal treats the problem by providing layered translations. If a page exists in a local language, it will automatically be displayed. If not, it will be displayed in the default company language which is called the **Corporate Language**.

The advantage of the system is that multilingual users can toggle back and forth between all available languages and see what their colleagues are saying about the same subject in their own countries — a true "think global, act local" solution. Best of all, it all happens automatically, which takes a big strain off the technical and content administrators when rolling out localized sites.

To explain the multi language behavior we will introduce the following terms:

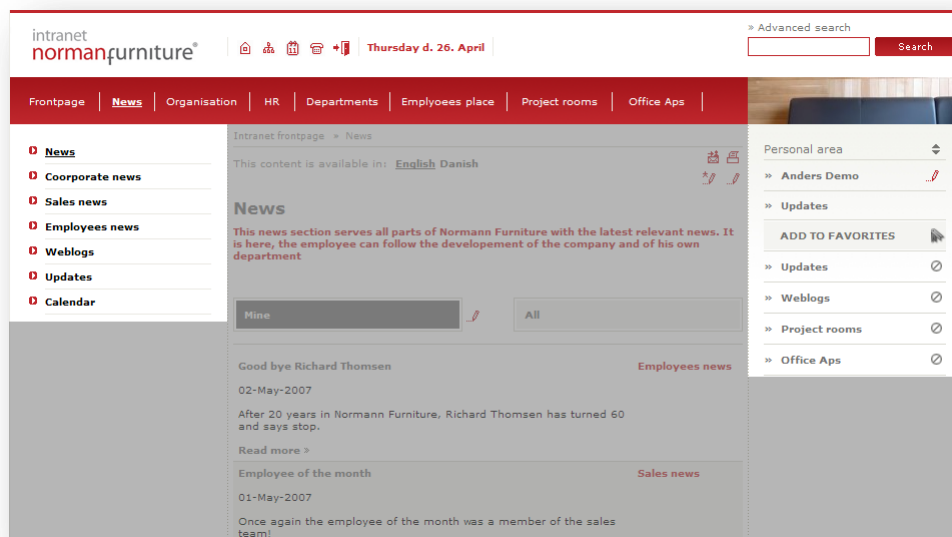
Term	Description	Image
Selected content language	The content language selected by the user on page. The language is reset to user preferred language when the user navigates to any other item.	
User preferred language	The language which users set in their profiles	
Corporate language	The company's corporate language. Set in the CorporateLanguage field in the following item: /Intranet/Site config/Settings.	
Website language	The default language of the site set in web.config file: <code><site ... language="en" ...></site></code>	

8.2 Content Area vs. Menu Area

The multilanguage support in Sitecore Intranet Portal applies to two different logical areas of the site:

- The content area
- The menu area, including the personal area

The location of these two areas is shown in the following image:



The menu area includes the top bar with navigation buttons and search, the top and left menus and personal area on the right. The content area (darkened in the image) contains the content itself – the news, announcements, blog posts, and so on

8.2.1 Language Behavior in the Menu Area

The algorithm for resolving languages in the menu area (everything outside the content area) is:

- User preferred language
- Corporate language
- Website language

This algorithm is used when deciding which *Skin* and *MainSettings* versions to use. In other words, if the *Skin* has no version in the user preferred language, the version in the corporate language is used. If no such version exist either, the version in the Website language is used.

For menu items, only the user preferred language is taken into account. If no version exists in this language, the item will not be displayed in the menu (unless it has a non-empty *CorporateTitle* field, as described later in this document).

8.2.2 Language Behaviour in the Content Area

The algorithm of the resolving languages in the menu area is:

- User selected content language
- User preferred language

- Corporate language
- Website language

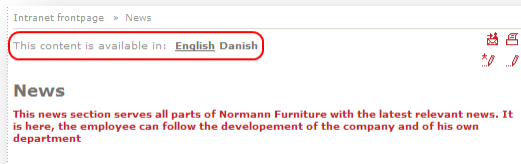
This algorithm is used for all content displayed inside the so-called localized placeholder, as well as content rendered inside the `<sc:contentArea>` tag (as described later in this document).

8.3 Working with Different Languages

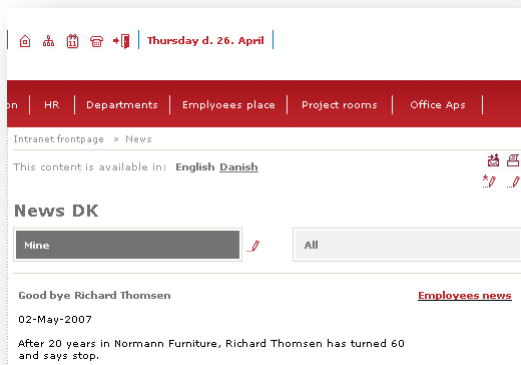
This section describes features related to working with different languages, such as the language selector, corporate language and so on.

8.3.1 The Language Selector

The content area has one distinctive feature — the language selector:




The currently active content language is underlined (English in this example). Select the Danish title to see the content of the current page in Danish:



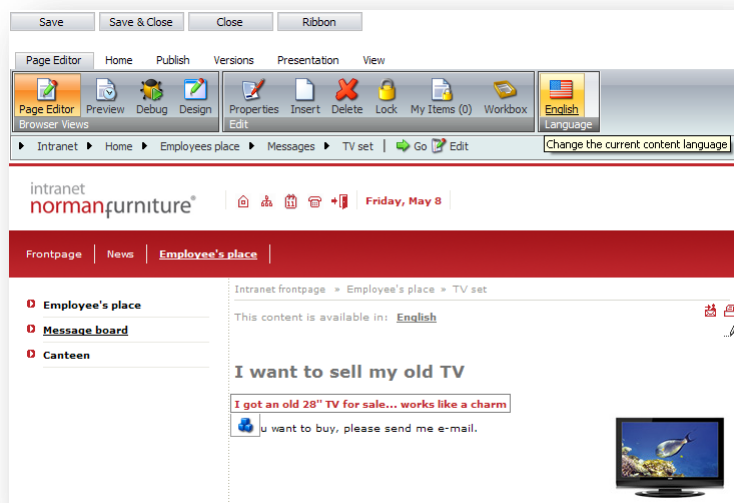
You can see that only the content language has changed to Danish, and the shell elements, such as menu titles, dates and navigation controls are still in English.

The selected content language is also used for the children of the current item if any. For example, the list of news will be rendered in Danish, if the user has selected the Danish language for the parent News item, provided that each news item has a version in Danish or a non-empty *CorporateTitle* field.

8.3.2 Editing Multilanguage Content

When a user clicks Edit  the **Page Editor** opens the version in the currently active content language.

In the editing mode it is possible to switch between languages:



The **Page Editor** renders the item in selected language.

Note

Branch templates can be given a language-specific name by specifying a **Display Name**. The display name will automatically be removed by the `IntranetAddVersionEventHandler` if set (but only if this is the first version created in the current language).

8.3.3 Corporate Language and CorporateTitle field

Sitecore Intranet allows you to specify a so-called corporate language. The corporate language is used as a fallback language if the content in the user's preferred language is not available.

The corporate language is used in combination with the *CorporateTitle* field that most of the standard intranet templates has. The *CorporateTitle* is used for displaying the menu item if an item doesn't have a title in the selected content language, thereby making the content available to all employees without having to translate/copy it to each individual language.

Typically, *Title* is the text in the current language, while *CorporateTitle* is in the corporate language. So when editing content in the corporate language, the text will often be the same in both fields (as illustrated in the previous screenshot).

In order to illustrate how the *CorporateTitle* works, consider the following scenario: The user's preferred language is Danish and the corporate language is English. A news item that exists in English and German will by default not shown to the user. But if the news item has a non-empty *CorporateTitle*, it will be shown to the user. In this case, if the user clicks the news item, the English version will be displayed (since the corporate language is English), and the user can click German in the language selector to see the German version.

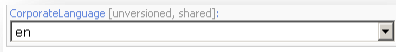
If an item has a non-empty *CorporateTitle* but is not available in either the preferred language or the corporate language, a message will be displayed that the content is not available in the selected content language and the user should select the language version manually:

Note

There is a `sc:menutitle()` XslHelper available, and the `IntranetItem` class now has a `MenuItem` property that can be used in sublayouts that use data binding. It will return the *Title* (if set) and the *CorporateTitle* as a fallback.

Changing the Corporate Language

The corporate language is set in the intranet settings under `/content/Intranet/Siteconfig/settings` in the field `CorporateLanguage`:



8.3.4 Adding a CorporateTitle Field to Custom Templates

Most items in the Sitecore Intranet site have a `CorporateTitle` field.

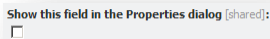
If you create custom templates for the solution, consider to add a `CorporateTitle` field to enable the associated functionality described above.

Notice that the `CorporateTitle` field must be a *Shared field*. If the field should be available in the Page Editor, also remember to select the **Show this field in the Properties dialog** checkbox.

8.3.5 Toggle Visibility of CorporateTitle for Specific Templates

In a default installation of the Sitecore Intranet, the field `CorporateTitle` is enabled on all the standard templates. This means that any employee can fill out `CorporateTitle`.

If you want to make the field available only to users that have access to the **Content Editor**, clear the template **Show this field in the Properties dialog** check box:



8.4 Working with the Localized Placeholder

To support that the language in the content area can be different than the language used for the menu area, Sitecore Intranet Portal uses a so-called "localized placeholder" in its layouts.

All renderings that are added to this placeholder will use the currently selected content language, while all other renderings (menus and so on) by default will use the user preferred language.

The localized placeholder overrides the standard Sitecore Placeholder Web control and takes care of switching the language. So if you create new layouts where you want to take advantage of this functionality, you should simply add the `LocalizedPlaceholder` control to your layout. An example of this can be seen in the `Intranet.Main.aspx` layout. An intranet assembly is registered and the control `LocalizedPlaceholder` is used instead of the `Normal` placeholder:

```
<%@ Register TagPrefix="intranetOverride"
Namespace="Sitecore.Intranet.HtmlControls" Assembly="Sitecore.Intranet" %>
```

```
<intranetOverride:LocalizedPlaceholder runat="server" ID="plhContent"
Key="content"></intranetOverride:LocalizedPlaceholder>
```

After adding the placeholder, you can refer to it like any other placeholder in a Sitecore layout.

Also notice that you can use the standard Sitecore XSLT functions and elements such as `sc:fld()`, `<sc:text>` and so on. They have been overridden in Sitecore Intranet Portal so that they work as expected in relation to the above description of how multilanguage support works in Sitecore Intranet Portal.

8.4.1 Using the LanguageResolver Outside the Localized Placeholder

If any of the renderings outside the localized placeholder display information from the current item, you must make sure that the content is displayed in the current content language. Otherwise the renderings use the interface language, which can result in wrong or unexpected content to be displayed.

The easiest way to switch to the content language is to add a `<sc:contentArea>` element in your rendering that surrounds all the relevant code:

```
<xsl:template match="*" mode="main">
  <sc:contentArea>
    <sc:html field="context" />
  </sc:contentArea>
</xsl:template>
```

Take a look at `Intranet.Context.xslt` or `Intranet.Author.xslt` for an example of renderings that use `<sc:contentArea>`.

Notice that you must still use `<xsl:if test="sc:menutitle(.)!=''">` to test that the items inside the `<sc:contentArea>` tag has a menu title. Also, you should always use `<xsl:value-of select="sc:menutitle(.)" />` inside the `<sc:contentArea>` tag to make sure that you take *CorporateTitle* into account when rendering out menu titles.

8.5 Adding and Removing Languages

This section describes how to add a new language, remove the English language as well as how to remove the language selector.

8.5.1 Adding a New Language

You can add new languages to Sitecore Intranet Portal in the same way as you add them to a common Sitecore solution.

Read about adding new languages to Sitecore solution on the [Sitecore Developers Network](#).

8.5.2 Removing the English Language

You must not delete the English language. This might cause Sitecore Intranet Portal nonworking.

Alternatively, you can make the English language node hidden. This will prevent it from being displayed in the language selector and the Page Editor.

If you use only one language, you can also consider removing the language selectors, as described further in this chapter.

8.5.3 Removing the Language Selectors

If you use only one language, you can remove the language selectors from the portal.

To remove the language selector, delete or comment the following tags in the `Intranet.Main.aspx`:

```
<cl:ContentLanguages runat="server"
ID="contentLanguages"></cl:ContentLanguages>
<cl:EditorLanguages runat="server"
id="editorLanguages1"></cl:EditorLanguages>
```

Deny the **Read** right on the following items or delete them at all (in the Core database):

```
/sitecore/content/Applications/WebEdit/Ribbons/WebEdit/Page
Editor/ContentLanguage
/sitecore/content/Applications/WebEdit/Ribbons/WebEdit/Versions/Language
```

8.6 Security and Languages

Standard Sitecore security can be used to control access to the different languages.

To read the content of the item the user must have (or inherit) the following access rights:

- **Read** right on the specific item
- **Read** and **Language Read** rights on the language item

To modify the item content a user must have (or inherit) the following access rights:

- **Read** and **Write** rights on the specific item
- **Language Read** and **Language Write** rights on the language item

Assign rights to language items under `/system/Languages`. If the **Languages** item is protected it needs to be unprotected before the rights can be set.

Chapter 9

Data Channeling

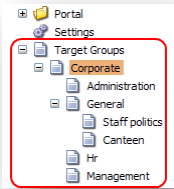
The Data Channeling Framework (DCF) is used to deliver specific content to specific target groups. While most people probably wants to know what is being served in the canteen today, not all people needs to know how a specific HR project is going. This information should probably only be targeted at the HR department. In Sitecore Intranet terminology this is called data channeling.

This chapter contains the following sections:

- Target Groups
- Filtering
- How to Disable the Data Channel Framework

9.1 Target Groups

The content covered by the framework is categorized into a number of target groups. Target groups can be nested one inside another. The available target groups are created as a hierarchy of the content items in Sitecore.



- The target group hierarchy can be created anywhere in the content tree. The default location is `/sitecore/content/Target Groups`.
- The target group is normally based on the `Intranet.TargetGroup` template, but target group items can actually be based on any template that has `Title` field.

9.1.1 Setting Security for Target Groups

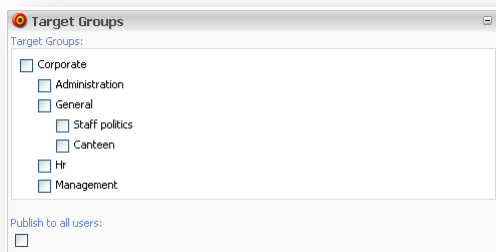
Notice that it is possible to apply standard Sitecore security (read/write permissions) to target group items to specify which users have access to a given part of the target group hierarchy.

- If the current user has **Read + Write** access to a target group (that is: to the content item that represents the target group), the appropriate checkbox is displayed.
- If the user has **Read** access (but not write access), a non-accessible check box is displayed.
- If one or more target groups are stored in the field and the current user does not have read access to some of these items, these target groups still remain selected when saving (in other words: although the current user cannot see these target groups, their state is not "lost" when saving)

9.1.2 Tagging Content with Target Groups

The content can be tagged with the target groups in either the Page Editor or the Sitecore client interface.

By default, the target group fields are not visible to users of the Page Editor, since this is typically an administrative task that is performed using the **Content Editor**:

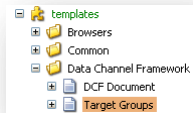


9.1.3 Setting Up Templates to Use the Data Channel Framework

Content that should take advantage of the data channeling framework will normally inherit from the `template /templates/Data Channel Framework/Target Groups`.

The template adds the following fields:

- **TargetGroups**
A tree like selector field that enables the author to select one or more target groups or hierarchy of target groups. The source must point the root item containing the target groups (by default `/sitecore/content/Target Groups`).
- **IgnoreTargetGroups**
if this field is selected then it makes sure that the content is targeted to all users.



None of the templates supplied with Sitecore Intranet inherit from the `Target Groups` template, but instead define the two fields themselves. This solution allows you to configure security and Page Editor settings separately for each template.

9.2 Filtering

This section describes how to implement filtering on the intranet portal.

9.2.1 Adding/Removing the Filtering Control on Lists

When you display lists to the user, for example news, the user can filter the list based on target groups or to show all the items in the list.



To either add or remove this filtering control, the following changes must be made.

Remove or add the `Intranet DataChannelControl` sublayout to the Default device of the template, and put it on the placeholder (typically `content` to add the filtering control to the content area of the page).



See how this is implemented in the `/templates/Intranet/Intranet.NewsSection` template.

If this sublayout is removed then filtering control that allows the user to choose between his targeted content and all content is no longer displayed. The data channeling framework is still working and now only the targeted content is being displayed.

To display all the content and bypass (or enable) the data channel framework you must also remove or add a `CanDisplay()` test to the rendering. The following example is from the `Intranet.NewsList.xslt` rendering. Here the `dcf:CanDisplay(.)` filters based on the user's target groups:

```
<xsl:template match="*" mode="main">
  <div class="NewsListContainer">
    <xsl:for-each select="./item[sc:fld('title',.) !='' and
@template='intranet.newsitem' and dcf:CanDisplay(.)] ">
```

9.2.2 Using DCF for Custom Renderings and Sublayouts

The previous scenario showed how to add or remove filtering on lists. In this scenario a XSL helper `dcf:CanDisplay(.)` is used to filter on the target groups that the current user is member of.

Another scenario is to do the filtering in a sublayout using the API. In that case, you use the static method

```
Sitecore.Intranet.DataChanneling.DataChannelingManager.CanDisplay(Item
item):
```

```
DataChannelingManager.CanDisplay(Sitecore.Context.Item);
```

To use the data channeling framework for custom renderings or sublayouts, follow these steps:

1. Add the target group fields to the template or inherit from the `Target Groups` template

2. Add the filtering control sublayout `Intranet DataChannelControl` to the layout of the template.
3. Add the conditional logic by using either the `dcf:CanDisplay()` XSL helper or the static method `DataChannelingManager.CanDisplay(Item item)`.

The `CanDisplay` XSL helper uses the following algorithm:

The item will be displayed if:

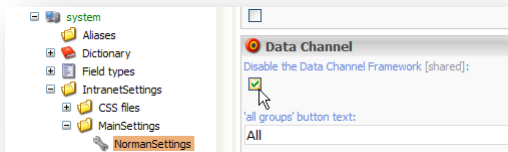
- The current user is not logged in (the function will return true).
- The item does not contain the *Target Groups* field or the field has no value (for example the item does not inherit from the `Target Groups` template).
- The item contains an *IgnoreTargetGroups* field and that field contains "1".
- The user has not selected any target groups to display (for example preview client is used and user is from 'sitecore' domain, not 'extranet').

If none of the above conditions are met, the target groups in the *Target Groups* field of the item are compared to the target groups that the current user has selected in one's user profile. If none of the target groups is present in both sets, the item will not be displayed.

9.3 How to Disable the Data Channel Framework

The Data Channel Framework can be disabled completely, so that the controls are not displayed on the portal (when editing items, editing user profiles - and also the filter controls will be hidden). When the Data Channel Framework is disabled, the `dcf:CanDisplay()` XSL helper method always return true.

To disable the Data Channel Framework select **Disable the Data Channel Framework** check box on the main settings item.



The controls will still be visible in the **Content Editor**, but will show no available target groups. To get rid of these fields, remove them from the relevant templates or hide them by setting field security.

Chapter 10

File Drop Area

Users can attach any type of file to their intranet page. Sitecore Intranet Portal includes a number of attachment controls, which use Sitecore's *File Drop Area* field. A Sitecore intranet administrator can use the *File Drop Area* section to configure the appearance of the attachments section. Moreover, administrators can modify the attachments views provided by default to meet specific user requirements.

The *File Drop Area* field uses WebDAV to let users manage files using the drag-and-drop feature. For more information about WebDAV in Sitecore, see [WebDAV Configuration](#) documentation on the [Sitecore Developer Network](#).

This chapter contains the following sections:

- Attachments and the File Drop Area

10.1 Attachments and the File Drop Area

This section describes how to enable the *File Drop Area* field on the intranet, how to modify the attachment view, and how the *Fallbackview* parameter works.

Note

If you are using the Drag & Drop or Attachments features and are logged in as a domain user, you must have access to the domain controller for that domain. This usually means that you need to be logged on to the same network as the controller. If you do not have access to the domain controller, the Drag & Drop and Attachments features will not function correctly.

10.1.1 Enabling the File Drop Area Field

To use attachments on an intranet page you should configure the item's template. To display attachments, the item's template must inherit the *File Drop Area Section* template. By default most intranet templates inherit the *File Drop Area Section* template.

To ensure that a template inherits the *File Drop Area Section* template:

1. In the **Content Editor**, navigate to the `Sitecore/Templates/Intranet` folder and select the template you are interested in.
2. On the **Content** tab, in the **__Base template** field, add the *File Drop Area Section* template to the *Selected* list.
3. Save the changes.

10.1.2 Modifying the Views

An attachment view determines how the attached files are displayed on an intranet page. Sitecore Intranet Portal contains six attachment views by default. These are located under the `Sitecore/System/IntranetSettings/File Drop Area/Views` folder. You can customize existing views, create new ones, and set the attachment view for the page.

To edit an existing view:

1. In the **Content Editor**, browse to `Sitecore/System/IntranetSettings/File Drop Area/Views`.
2. Select the view you want to modify.
3. In the **Layout** field, see which `.ascx` file defines this view and edit it as you wish.

To add a new view:

1. In the **Content Editor**, duplicate an existing view item under the `Sitecore/System/IntranetSettings/File Drop Area/Views` folder.
2. Create a new `.ascx` file and specify it in the **Layout** field.
3. Specify the placeholder where you want to use the new layout.

To change the Attachments view of the page:

1. In the **Content Editor**, navigate to the page item.
2. In the **File Drop Area** group, in the **File Drop Area View** drop-down list, select the view.

10.1.3 Changing the Drag and Drop Window

Sitecore Intranet Portal users manage attachments in the **Drag and Drop** window. There are two types of **Drag and Drop** window in Sitecore Intranet Portal:

- Sitecore dialog
- Windows Explorer window

The main difference between these windows is that the Windows Explorer window opens faster (especially at connections with high ping times) but the user must refresh the page to display the added attachments.

You can configure Intranet Portal to use a specific **Drag and Drop** window by editing the `Intranet.OpenAttachmentsInExplorer` parameter in the `web.config` file:

- True — attachments are managed in the Windows Explorer window.
- False — attachments are managed in the Sitecore dialog.

The `Intranet.OpenAttachmentsInExplorer` parameter is set to false by default.

10.1.4 The Fallbackview Parameter

When working with attachments users' browsers are divided into two groups:

- Browsers which support WebDAV (Internet Explorer)
- Browsers which do not support WebDAV (for example Mozilla Firefox)

If the attachments on an intranet page use a view which requires WebDAV, the browsers that do not support WebDAV will not be able to use all the features in the Attachments control when you are in Edit mode. Use the *Fallbackview* parameter to specify which alternative view should be used in such cases. Of the default views, the Inline Icons and Descriptions Content View is the only view that has the *Fallbackview* parameter because this is the only view that renders a WebDAV control straight on an intranet page.

To set the *Fallbackview* parameter for an attachments view:

1. In the **Content Editor**, browse to the `Sitecore/System/IntranetSettings/File Drop Area/Views` folder and select the view item.
2. Set the *Fallbackview* parameter and save the changes.

Chapter 11

RSS Feeds

Sitecore Intranet Portal supports RSS feeds. This chapter describes how to configure RSS feeds on SIP.

If your Sitecore Intranet Portal solution uses the Active Directory (AD) module, you must be using the AD module version 1.0.2 rev.090914 or later.

This chapter contains the following sections:

- Disabling RSS Feeds on an Intranet
- Modifying Intranet Portal RSS Feeds
- Content Item Feeds

11.1 Disabling RSS Feeds on an Intranet Site

RSS feeds are enabled in Sitecore Intranet Portal by default.

To disable the RSS feeds on all the pages on the intranet site:

1. In the **Content Editor**, navigate to the `Website/App_Config/Include` folder.
2. In the `SIP_RSS.config` file set the `PortalRSSFeeds.Enabled` parameter to `false`.

If you disable the RSS feeds, the RSS feed icons are not displayed on the site but subscribers will still receive the feeds to which they have previously subscribed.

11.2 Modifying Intranet Portal RSS Feeds

Sitecore Intranet Portal RSS feeds are predefined for every intranet page (except for the front page). A SIP administrator can configure how intranet pages are presented in the RSS Feed.

To configure how an RSS feed is presented:

1. In the **Content Editor**, select the item that you want to configure the RSS feed for.
2. On the **Presentation** tab, in the **Feeds** group, click **Design**.
3. In the **Feed Presentation** dialog box, specify how the feed is presented to the user who subscribes to this RSS feed.

Field Name	Description	Default Value
Title	The name that you want to appear in the title of the feed.	The Title field of the item
Body	The text that you want to appear in the body of the feed.	The Feed Entry Content field of the item.
Date	The date that you want to appear in the feed. This could be, for example, the date the item was created, updated, or the date from which the news applies. In the Updates feeds the date when the item was updated is always displayed.	The Created field of the item
Preview	A preview of the feed.	

4. When you have filled in these fields, click **OK**.
In the **Feeds** group, you can see that the **Design** button now has a green check mark indicating that an RSS feed has been designed for this item.
5. In the **Content Editor**, save your changes.

Note

This item and all the other items based on this template are presented in the same way in the feed.

To modify RSS feeds for items based on different templates repeat this procedure for each different template.

Note

An RSS feed contains updates for a page and all of its sub-pages.

11.2.1 The Feed Entry Content Field

The **Feed Entry Content** field displays several item fields in the body of the RSS Feed. This field can be customized. This value can be set for an individual item, an item template, and for all the templates that inherit the `RSS Feed Entry Content` template. Use tokens to configure how information is displayed in this field. Tokens are keywords that are replaced with the appropriate field values. You can also apply rich text formatting to these tokens.

The format of these tokens is: `[field_name]`.

Note

Tokens cannot contain spaces between square brackets and a field name. For example, `[title]`, `[body]` are valid values, whereas `[title]`, `[body]` are invalid values.

To find the correct field name to use in the token:

1. In the **Content Editor**, navigate to any item.
2. In the **Quick Info** section, click the template link.

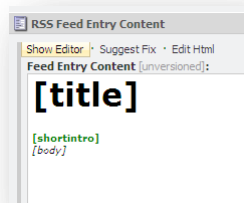
The **Template Manager** window displays a list of template fields.

You can use any field as a token. Just copy and paste the field name when configuring the **Feed Entry Content** field.

To configure the **Feed Entry Content** field in the RSS Feed Entry Content template:

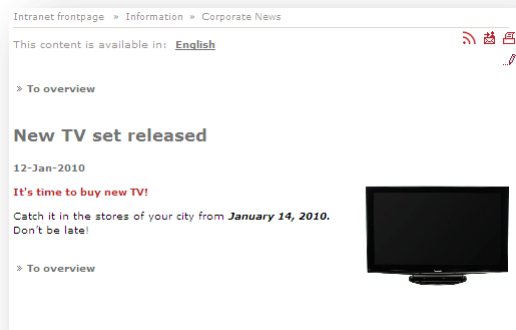
1. In the **Content Editor**, navigate to the Sitecore/templates/intranet/configuration/RSS Feed Entry Content/___Standard Values item.
2. Select the **RSS Feed Entry Content** field and click **Show Editor**.
3. Edit the field.

In this example we use the [title], [shortintro] and [body] tokens:



4. Save the changes.

This is how the intranet page looks:



This is how the RSS feed for this page looks:



11.3 Content Item Feeds

In Sitecore Intranet Portal you can use a content item RSS feed to ensure that you are informed of any changes that are made to the item. The content item feed contains details about the version history of the item, validation information, workflow history, and so on. The content item feed looks like this:

The New TV set released was moved to the Draft workflow state

Updates to New TV set released Sitecore item

Posted On: Tue 1/26/2010 6:11 PM

The New TV set released item was moved from to Draft workflow state by Administrator

[/Intranet/Home/Test/Corporate News/New TV set released](#), English, Version 1

What do you want to do with New TV set released:

- [Edit it](#)

Validation information:

Warning The field "News image" contains broken links

Warning This item contains broken links in: News image

Warning The item name contains characters that will be encoded when used in a link.

[Preview the webpage](#)

Workflow history:

Date	User	Previous State	Current State	Comment
26. januar 2010 18:10	sitecore\admin		Draft	Item created

Note

Sitecore Intranet Portal provides users with different types of feeds: content item feeds, intranet portal feeds, workflow and workflow state feeds. For more information about the latter three feed types, see the *SIP User Guide, section 4.2, RSS Feeds* on the SDN.

11.3.1 Subscribing to a Content Item RSS Feed

In the **Content Editor**, you can subscribe to a feed for any item. However the feed will only contain useful information if the item is subject to a workflow. When you subscribe to a content item RSS feed, you can submit, approve, or reject an item version and edit it as long as you have the appropriate permissions.

To subscribe to a content item feed:

1. In the **Content Editor**, navigate to the item that you are interested in.
2. Click the **Review** tab, and then in the **Proofing** group, click **Subscribe**.

To subscribe to all sub-pages click **Subscribe to All Subitems**.

3. In the Web page that opens, click **Subscribe to this feed** and add this RSS feed to your favorites in Internet Explorer.

- If you use another RSS Reader (for example Microsoft Outlook), copy the URL of the Web page and paste it into your RSS Reader.

The New TV Released was moved to the Draft workflow state

Wednesday, March 03, 2010, 1:18:02 PM

The New TV Released item was moved from to Draft workflow state by [Thomas Rebo](#)

[/intranet/Home/Test/Corporate News/New TV Released](#)
Danish, Version 1

What do you want to do with New TV Released:

- [Publish](#) or [Publish & comment](#)
- [Edit it](#)

Validation information:

Warning The item name contains characters that will be encoded when used in a link.

[Preview the webpage](#)

Workflow history:

Date	User	Previous State	Current State	Comment
3. marts 2010 13:18	t@t		Draft	Item created

Using the Content Item RSS Feed

The Content Item RSS Feed contains information about the item, including:

- The name of the item.
- Its current workflow state.
- The workflow history of the item.
- A table showing the differences between the new version and the old version of the item.

When you have subscribed to a content item feed, you can use RSS feed to perform the tasks that you have permission for. The list of commands that you can perform also depends on a workflow that is assigned to the item you have subscribed to. For example, you can do the following:

- Approve or approve and add a comment.
- Reject or reject and add a comment.
- Submit or submit and add a comment.
- Publish or publish and add a comment.
- Edit the item.

Chapter 12

Draft Mode

Sitecore Intranet Portal allows users to save draft pages. This feature is called Draft Mode. This chapter describes the main principles of Draft Mode.

This chapter contains the following sections:

- Draft Versions
- Enabling Draft Mode
- Creating a Draft Workflow

12.1 Draft Versions

SIP supports draft versions. This allows users to save an item without publishing it on the intranet and these items are only visible to the author. A draft version is created when you edit an existing item that has been published and has been assigned to a draft workflow. Similarly, a draft version is also created when you create a new item which is based on a template that has been assigned a draft workflow,

SIP only creates the new version when you click **Save**. When the user is finished editing the item, they can either click **Publish** to publish the item on the intranet and move it to the final workflow state or click **Discard** to delete this version.

Note

A new version is only created for items which have already been published – that is to say, items that are in the final workflow state.

Note

Auto-versioning will not function for the draft workflow. To read more about auto versioning, see [SIP Main Settings on the SDN](#).

12.1.1 The Draft Workflow

SIP contains the Draft Workflow by default.

The Draft workflow contains two workflow states:

- *Draft* (the item is unpublished)
- *Final* (the item is published).

You can configure any number of draft workflows.

The `Use Draft Mode Workflow` base template has been assigned the Draft workflow by default. Any templates which use the `Use Draft Mode Workflow` as a base template will therefore have the Draft workflow enabled. If the item you are editing is based on a template that already has a workflow assigned to it, this workflow will be used instead of the Draft workflow.

Note

If a workflow has more than two states, all of the states before the final one are “draft” states. This means that items in those states are not published on the intranet and SIP does not create a new version when you edit them.

12.1.2 Who is the Author of the Draft Version

The author of the draft version is the only person who can see and edit this version in the **Page Editor**. Even if an author shares the draft version with another user, for example their manager, and the manager edits the item, the original author will still be the owner of the item, until that item is published.

SIP uses the following priorities to determine who is the author of a draft version:

1. The user who has locked the item.
2. The user specified in the Owner field of the item Security section.
3. The user who has updated the item.
4. The user who has created the item.

12.2 Enabling Draft Mode

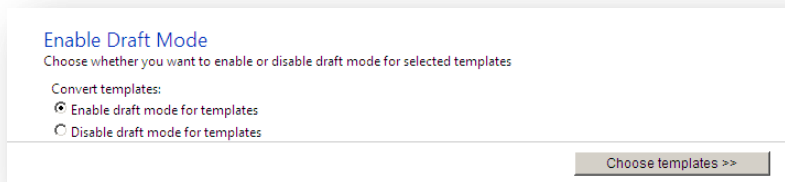
You use the **Enable Draft Mode** tool to assign the Draft workflow to templates. The `Use Draft Mode Workflow` template in SIP contains the Draft Workflow in the `__Standard Values`. The **Enable Draft Mode** tool adds the `Use Draft Mode Workflow` template as the base template to the selected templates. As the result all the selected templates use the Draft workflow by default.

Note

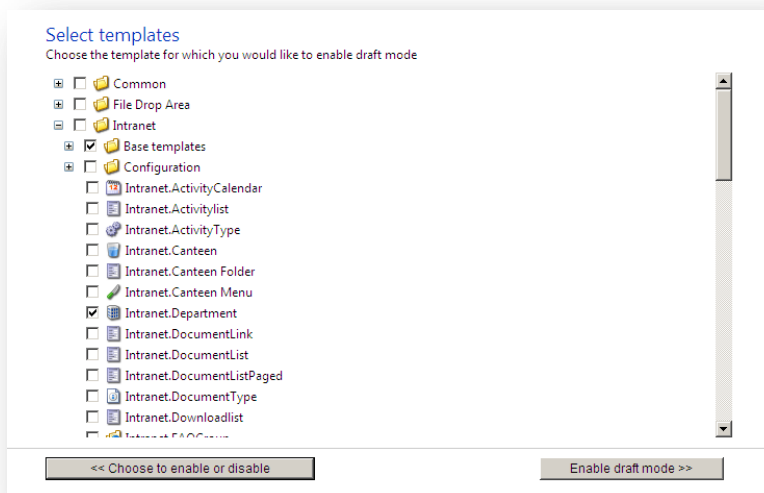
To use the **Enable Draft Mode** tool the user must have SIP administrator privileges.

To enable Draft Mode for a template:

1. In your browser's address bar type:
`http://localhost/sitecore/admin/enabledraftmode.aspx`
2. On the *Log into Sitecore* page, type the username (domain\user_name) and password.
3. Click **Login**.
If you select the **Remember me** check box while logging in, you can skip this page in the future.
4. On the **Enable Draft Mode** page, select the **Enable draft mode for templates** option.



5. On the **Select templates** page, select the templates that you want to enable draft mode for. To select all the templates in a folder, select the folder.



6. Click **Enable draft mode** to finish.

View the result message on the **Results** page.

To disable Draft Mode for a template, repeat this procedure and in step three, click **Disable draft mode for templates**.

12.3 Creating a Draft Workflow

As mentioned earlier, you can create any number of draft workflows.

To create a draft workflow:

1. In the **Content Editor**, create a new workflow under the *Sitecore/system/workflows* item.
2. In the new workflow item, in the **Quick Info** section, copy the value in the **Item ID** field.
3. In the `web.config` file, find the `Intranet.DraftWorkflows` setting and paste the Item ID value of the new workflow.

If more than one user needs to be able to edit an item in this workflow, for example, a content editor and their manager who has to approve the item, you must configure the *Owner* field to ensure that it always reflects the current author.

To configure the owner field so that it supports multiple users, create a `SetOwnerAction` command for the Draft workflow. The `SetOwnerAction` command defines the *Owner* field. The `SetOwnerAction` command is defined in the `Sitecore.Intranet.dll` file in the *Website/bin* folder.

Chapter 13

Other Features

This chapter describes how to convert Rich Text fields to Word fields, and vice-versa.

This chapter contains the following sections:

- Converting between the Rich Text and Word Field Types

13.1 Converting between the Rich Text and Word Field Types

Sitecore Intranet Portal can automatically convert fields from the Rich Text field type to the Word field type, and vice-versa.

To convert from the Rich Text field type to the Word field type:

1. In your browser's address bar type:
`http://localhost/sitecore/admin/convertwordfields.aspx`
2. On the *Log into Sitecore* page input the username (domain\user_name) and password. Click **Login**.
If you select the **Remember me** check box while logging in, you can skip this page in the future.
3. On the **Convert Word Document fields** page select **From Rich Text to Word**.
4. On the **Select fields** page, select the fields that you want to convert.
To choose all the fields in a folder or in a template, select this folder or template. Click **Convert the selected fields**.
5. View the result message on the **Results** page.

To convert fields from the Word type to the Rich Text type, repeat this procedure and in step two, select **From Word to Rich Text** option.

Chapter 14

Developer's Notes

This chapter describes useful information for Sitecore Intranet Portal developers.

This chapter contains the following sections:

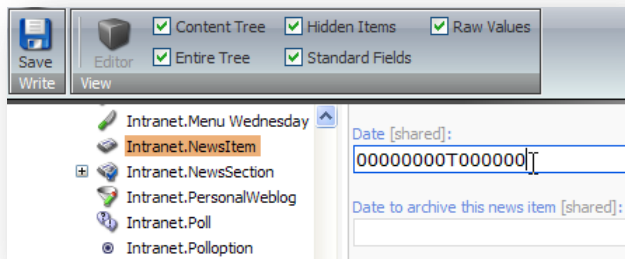
- Dates
- XSLT
- Styles
- Search
- Draft Mode
- Side Menu

14.1 Dates

Sitecore Intranet Portal lets you set dates automatically as well associate dates with user's current language.

14.1.1 Automatically Setting Date or Datetime Fields to Today's Date or Current Timestamp

Sitecore allows automatically setting *date* or *datetime* fields to today's date or current timestamp. To do this in the template put this string `00000000T000000` into the relevant *date* or *datetime* field (switch to view **Raw values** when doing this).



It is the `Sitecore.Intranet.EventHandlers.IntranetSetDateHandler` that substitutes the `00000000T000000` timestamp with the actual timestamp.

14.1.2 Date Formatting and Parsing

Dates are output and parsed using the format associated with the user's current language, not the content language. Utility functions are available in the `IntranetDateUtil` class and the `LocalizedXslHelper` class.

The places where date formats are specified are as follows:

- MainSettings: **ShortDateFormat**, **LongDateFormat**, **topdateformat**, **newsdateformat**, **newsdateformatlists**
- WebLogSettings: **datepattern**

The **ActivitySearchDateText** text has been changed to e.g. `{0}` to dynamically display a sample date using the correct date formats on the activity search page.

14.1.3 Changing Date Format

In Sitecore Intranet Portal, dates are displayed according to a regional standard. To change the date format:

1. In the **Content Editor**, select the `sitecore/system/Languages/language_name` item.
2. In the **Regional ISO Code**, enter the code. For example `en-US`.

14.2 XSLT

You can implement various functions with XSLT controls and extensions.

14.2.1 Methods to Retrieve the Most Important Items in XSLT

The `IntranetUtil` class contains the following methods `GetSiteRoot()`, `GetHomeItem()`, `GetMainSettingsItem()`, `GetSkinItem()`, and so on. By default these methods are used in the XSLT files resulting in better performance and a more consistent way of getting these references.

Note

The `IntranetSettings.GetSettingsItem(string)` method has been made public to make it easier to retrieve a reference to any custom settings items, such as the `CustomMainSettings` item.

14.2.2 XSLT Controls and Extensions

Isfieldvisible

This is a helper function. The format is the following:

`sc:isfieldvisible('fieldname', XPathNodeIterator)` — checks if the *fieldname* field of the item specified by `XPathNodeIterator` ('.' by default) should be rendered on the portal or not.

`sc:isfieldvisible('fieldname', XPathNodeIterator, 'subvalue')` — checks if the *subvalue* subvalue of the *fieldname* field of the item specified by `XPathNodeIterator` ('.' by default) should be rendered on the portal or not.

Longdate

This xslt control outputs dates in SIP long date format. This format is defined in *SIP Main Settings* item, *Long date format* field. You can set different date formats for different languages.

Using this xslt control allows changing date field using the **Page Editor**.

The xslt control format is the following:

`<sc:longdate field="fieldname" [select="XPathNodeIterator"]/>` — outputs the date from the *fieldname* field of the item specified by `XPathNodeIterator` ('.' by default);

`<sc:longdate select="iso date"/>` — outputs the date specified in the `select` attribute using ISO date format.

Shortdate

This xslt control outputs dates in SIP short date format. This format is defined in *SIP Main Settings* item, *ShortDateFormat* field. You can set different date formats for different languages.

Using this xslt control allows changing date field using the **Page Editor**.

The xslt control format is the following:

`<sc:shortdate field="fieldname" [select="XPathNodeIterator"]/>` — outputs the date from the *fieldname* field of the item specified by `XPathNodeIterator` ('.' by default);

`<sc:shortdate select="iso date"/>` — outputs the date specified in the `select` attribute using ISO date format.

14.3 Styles

You can configure the styles that are used to display content in the intranet portal.

14.3.1 Word Field Styles

Users edit content on the Web site using the Rich Text Editor or Microsoft Word. If you are using the Word field, you can configure how to display the content of the field using the `Intranet.AddWordStylesOnPage` parameter in the `web.config` file:

- **True** — the content is displayed using the same styles used by Microsoft Word. These styles may conflict with other styles used on the intranet portal but will be displayed as the user intended.
If the user did not specify the text font while editing the Word field Word's default font (Times New Roman) is used to display the text.
- **False** — the content is displayed using the default intranet content style instead of the default Microsoft Word styles. This means there may be a difference between what is entered by the user in the Word field and what is displayed on the site.

14.4 Search

You can configure search parameters on the intranet portal.

14.4.1 Customize Search Output

You can customize an output of search results:

1. Derive a new class from the `Intranet.Search.ResultsFormatting.SearchResultFormatter` class and override necessary methods. This class is used to render an output of search results.
2. Derive a new class from the `Intranet.Search.Layouts.IntranetSearch` class and override its `GetSearchResultFormatter` method to return your custom formatter.
3. In the `/layouts/Intranet.Search.ascx` file, in the `@ Control` directive, change the `Inherits` attribute to refer to your customized version of the `IntranetSearch` class.

14.5 Draft Mode

You can customize *Draft Mode* using the following classes:

- `IntranetItemProvider`
- `DraftModeDisabler`

14.5.1 `IntranetItemProvider` Class

Sitecore Intranet Portal uses its own item provider to support *Draft Mode*. The `IntranetItemProvider` class is responsible for the work with items. This class contains virtual methods that you can override to implement your custom *Draft Mode* logic.

14.5.2 `DraftModeDisabler` Class

The `DraftModeDisabler` class is used to disable *Draft Mode* for a task or feature. For example, we use this class to disable *Draft Mode* when rendering a list of item versions on the web page. If *Draft Mode* is enabled the list does not display an item version that is not published.

When instantiating a `DraftModeDisabler` class you can also disable the Filter Items functionality that is used to respect publishing restrictions when retrieving items.

To disable Filter Items in the `DraftModeDisable` pass `True` value to its constructor.

14.6 Side Menu

Sitecore Intranet Portal contains left hand side menu that lets users navigate the intranet. Side menu is implemented as Sitecore sublayout (ascx user control). You can configure Side Menu to add menu items. This might be useful to add a menu item referring to an external source.

To add a reference menu item, in the Content Tree create an item based on the `Intranet.DocumentLink` template and use the **Link to** field to add the reference.

Important

In the **Link to** field, the **Insert Javascript** and **Insert Anchor** buttons work but SIP ignores inserting Javascript links and anchors.

14.6.1 Adding Side Menu Item

To add a Side Menu item:

1. Add a new processor to the `GetMenuItems` pipeline.
2. This processor contains the `Process` method and gets a parameter of the `MenuItemArgs` type.
This processor must fill in the `MenuItem` property with the list of `SubMenuItem` objects corresponding to the menu items.