

Search Providers & Settings for Properties

Mari Jørgensen - EPiServer Norway

EPiSERVER
PARTNER
SUMMIT

June 3 – 6



Agenda

- » Search Providers
- » Settings for Properties

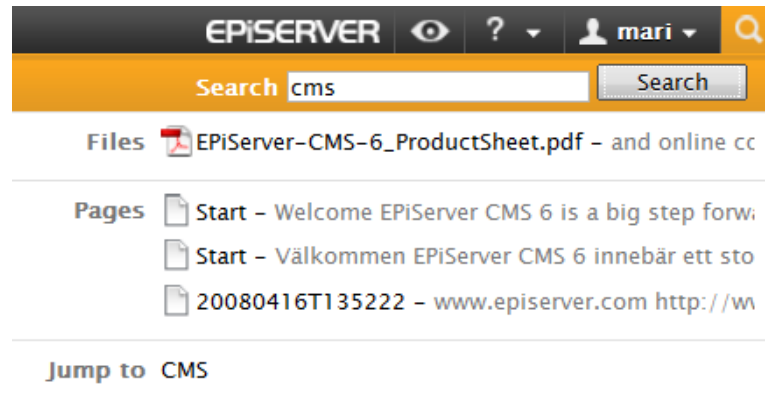
Search Providers

- » OnlineCenter features search functionality by aggregating results from several search providers.
- » Each provider reports results based on a specific category, e.g. Pages, Blogs, Files etc.

Search Providers

» EPiServer CMS 6 includes three search providers

- FileSearchProvider
- PageSearchProvider
- MenuSearchProvider



Search Providers - Demo


- » Creating a custom Search provider

EPiServer.Shell.Search.ISearchProvider


Implement interface to create a custom provider

```
public class UserSearch : ISearchProvider
```

Methods

	Name	Description
	Search	Executes a Search on the provider

Properties

	Name	Description
	Area	Area that the provider maps to, used for spotlight searching
	Category	The category that the provider returns hits in

System.ComponentModel.Composition.Export

```
[Export(typeof(ISearchProvider))]  
public class UserSearch : ISearchProvider
```

Managed Extensibility Framework is used to export dependencies from the modules inside the Online Center →

*You will need to add a reference to the MEF assembly:
System.ComponentModel.Composition*

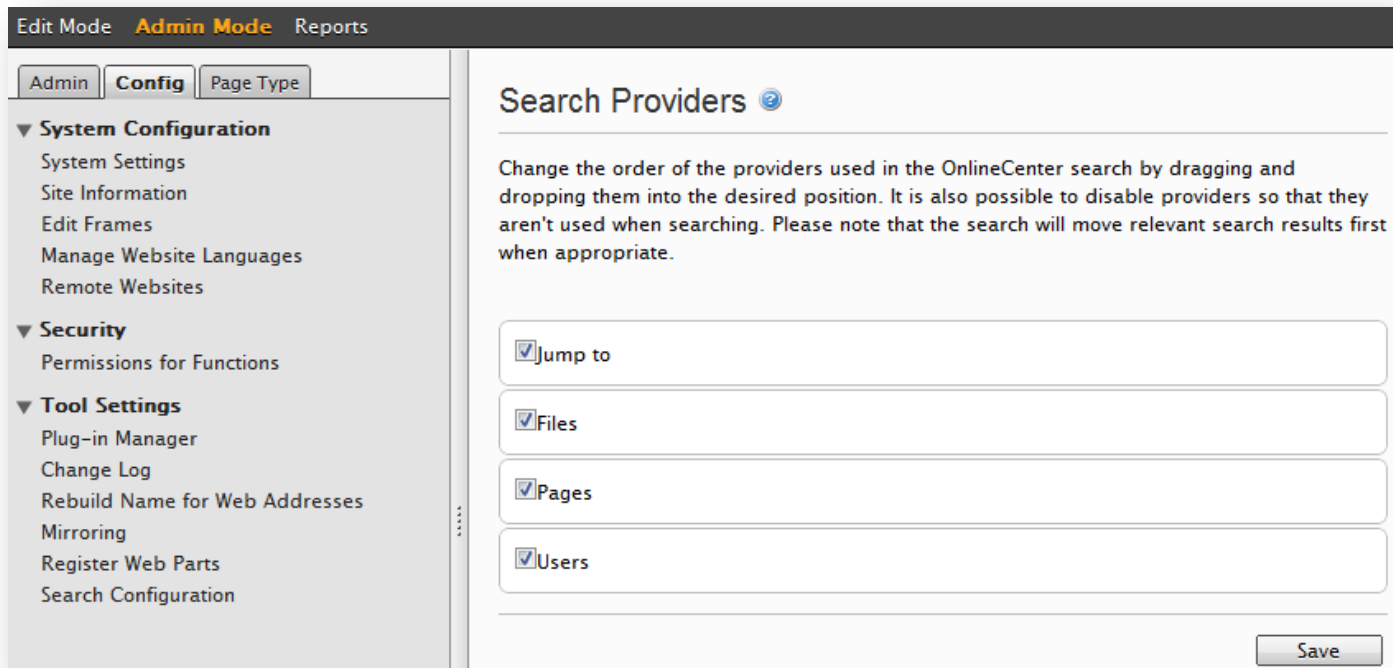
Add the Configuration

To add a custom search provider, you need to configure your assembly as a shell module.

```
<episerver.shell>
  <protectedModules rootPath="~/secure/">
    ...
    <add name="UserSearch">
      <assemblies>
        <add assembly="Demo.UserSearchProvider" />
      </assemblies>
    </add>
  </protectedModules>...
```


Search configuration

» Ordering can be changed in admin mode



The screenshot shows the 'Search Providers' configuration page in the 'Admin Mode' of a web application. The left sidebar contains a navigation menu with sections: 'System Configuration' (System Settings, Site Information, Edit Frames, Manage Website Languages, Remote Websites), 'Security' (Permissions for Functions), and 'Tool Settings' (Plug-in Manager, Change Log, Rebuild Name for Web Addresses, Mirroring, Register Web Parts, Search Configuration). The main content area is titled 'Search Providers' with a help icon. It contains a descriptive paragraph: 'Change the order of the providers used in the OnlineCenter search by dragging and dropping them into the desired position. It is also possible to disable providers so that they aren't used when searching. Please note that the search will move relevant search results first when appropriate.' Below this, there are four input fields, each with a checked checkbox and a label: 'Jump to', 'Files', 'Pages', and 'Users'. A 'Save' button is located at the bottom right of the main content area.

Edit Mode **Admin Mode** Reports

Admin **Config** Page Type

▼ **System Configuration**

- System Settings
- Site Information
- Edit Frames
- Manage Website Languages
- Remote Websites

▼ **Security**

- Permissions for Functions

▼ **Tool Settings**

- Plug-in Manager
- Change Log
- Rebuild Name for Web Addresses
- Mirroring
- Register Web Parts
- Search Configuration

Search Providers ?

Change the order of the providers used in the OnlineCenter search by dragging and dropping them into the desired position. It is also possible to disable providers so that they aren't used when searching. Please note that the search will move relevant search results first when appropriate.

☒ Jump to

☒ Files

☒ Pages

☒ Users

Save

References – Search Providers

- » [Online Center Developer Documentation](#)
- » [Blog: Implementing a Search provider for SiteCenter in 5 minutes](#)

Settings for Properties

- » Store custom settings for a property
 - Create and store settings for your custom properties
 - Editable through ui inside admin mode
- » Possible to define global and custom settings

Settings for Properties - Demo

Settings attribute and class

Add Settings using attribute PropertySettings

```
[PropertySettings(typeof(MySettings), true)]  
public class IntroControl : PropertyLongStringControlBase
```

The Settings class need to implement interface IPropertySettings, most easily accomplished inheriting from PropertySettingsBase. Attribute PropertySettingsUI defines the class that is used to edit the settings.

```
[PropertySettingsUI(AdminControl = typeof(MySettingsUI))]  
public class MySettings : PropertySettingsBase
```

Settings UI

```
/// <summary>  
/// Used to render a UI for editing the settings in the admin UI.  
/// Needs to be in the inherit from Control and implement  
/// IPropertySettingsUI.  
/// Most easily accomplished using PropertySettingsControlBase  
/// </summary>  
public class MySettingsUI : PropertySettingsControlBase
```

It's also possible to specify the url to a user control through either the `Url`, `UrlFromUi` or `UrlFromUtil` property. This user control needs to either implement `IPropertySettingsUI` or inherit `PropertySettingsUserControlBase`.

Accessing Settings object

PropertyData has got a new Property....

```
public PropertySettingsContainer SettingsContainer
```

... and a new method

```
public IPropertySettings GetSetting(Type settingsType)
```

Example of use

```
MySettings settings = (MySettings)  
    PropertyData.GetSetting(typeof MySettings))
```

References – Settings for Properties

- » <http://labs.episerver.com/en/Blogs/Linus-Ekstrom1/Dates/2009/10/Settings-for-properties/>



marijorg

EPISERVER
PARTNER
SUMMIT