

If the default widget dashboards in the administration interface do not meet your requirements, you can create your own dashboard applications or pages.



Important: Widget dashboards are a completely separate feature from the system's main [application dashboard](#). Every widget dashboard is either a standalone application, or a page within another application.

The following *example* demonstrates how to add a custom dashboard application to the **Social & Community** category. You can apply the same principles when creating widget dashboards in other locations.

Creating the dashboard page template

First, create a [page template](#) for the dashboard:

1. Open the **Page templates** application.
2. Select the **Dashboard pages** category in the tree.
3. Click **New template** and enter the following values:
 - **Template display name:** Community dashboard
 - **Template code name:** CommDashboard
4. Click **Save**.
5. Select *Dashboard page* as the **Template type**.
6. Click **Save**.

Adjust the [layout](#) of the page template:

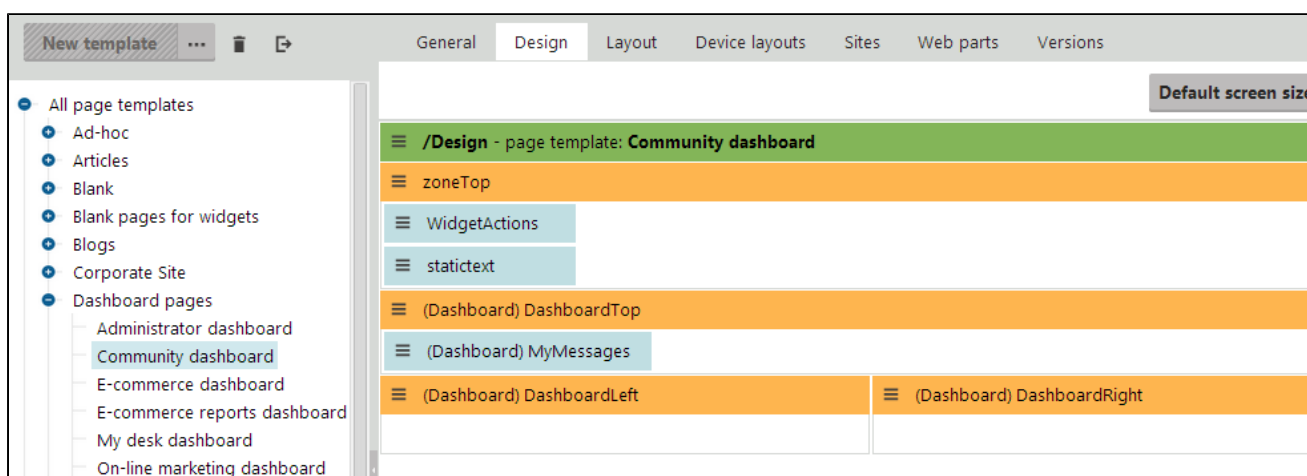
1. Switch to the **Layout** tab.
2. Copy the following sample code into the layout to define web part/widget zones for the template:

```
<table border="0" width="100%" cellpadding="0" cellspacing="0">
  <tr>
    <td colspan="2">
      <div class="DashboardActions PageTitleHeader">
        <cms:CMSWebPartZone ID="zoneTop" runat="server" />
      </div>
    </td>
  </tr>
  <tr>
    <td colspan="2">
      <cms:CMSWebPartZone ID="DashboardTop" runat="server" />
    </td>
  </tr>
  <tr valign="top">
    <td style="width:50%">
      <cms:CMSWebPartZone ID="DashboardLeft" runat="server" />
    </td>
    <td style="width:50%">
      <cms:CMSWebPartZone ID="DashboardRight" runat="server" />
    </td>
  </tr>
</table>
```

3. Click **Save**.
4. Switch to the **Design** tab.
5. Expand the menu (☰) of the **DashboardTop** zone and click **Configure**.
6. Switch the **Widget zone type** property from *None* to *Dashboard*.
7. Click **Save & Close**.
8. Repeat the steps 5 – 7 for the **DashboardLeft** and **DashboardRight** zones.

Add [web parts](#) and [widgets](#) to the zones:

1. **Add** the **Widget actions** web part into the **ZoneTop** zone.
2. Configure the following properties of the *Widget actions* web part:
 - **Widget zone type:** Dashboard
 - **Widget zone ID:** Leave empty (*Designates the zone where new widgets are created when users click the Add widget button. By default, the web part uses the first available zone (DashboardTop in this case), but you can specify the ID of any other dashboard zone.*)
3. Leave the remaining properties in their default state and click **Save & Close**.
4. Add the **Static text** web part to the same zone and set the following properties:
 - **Text:** This is a custom dashboard page
 - **Display as:** Header level 2
5. Click **Save & Close**.
6. Expand the menu (≡) of the **DashboardTop** zone and click **Add new widget**.
7. For example, choose the **Community -> My messages** widget.
8. Confirm the dialogs without making changes and leave the other two dashboard zones empty.
 - This sets the default content of the dashboard that individual users can later configure and expand.



Adding the dashboard UI element


To create a new widget dashboard application, you need to add a UI element to the system:

1. Open the **Modules** application.
2. Edit (✎) the **Custom** module. Note that the **Module code name** is *cms.customsystemmodule* on the **General** tab.
3. Switch to the **User interface** tab.
4. Select the **CMS -> Administration -> Social & Community** element in the tree.
5. Click **New element** (+).
6. Enter the following values:
 - **Display name:** Community overview
 - **Code name:** CommDashboardElement
 - **Module:** Custom
 - **Caption:** Community overview
 - **Element icon type:** Class
 - **Element icon CSS class:** icon-app-content-dashboard
 - **Type:** URL
 - **Target URL:** `~/CMSGlobalFiles/CommDashboard.aspx?dashboardName=Comm&templateName=CommDashboard&{hash}`
 Sets the URL of the page with the content of the UI element. You will create the source file used in the URL above later in the example. When creating links to dashboard pages, you need to understand and correctly specify the query string parameters:
 - **dashboardName** - sets a name for the dashboard to ensure uniqueness in cases where multiple dashboards use the same page template. The content of a dashboard is unique for every user. If two or

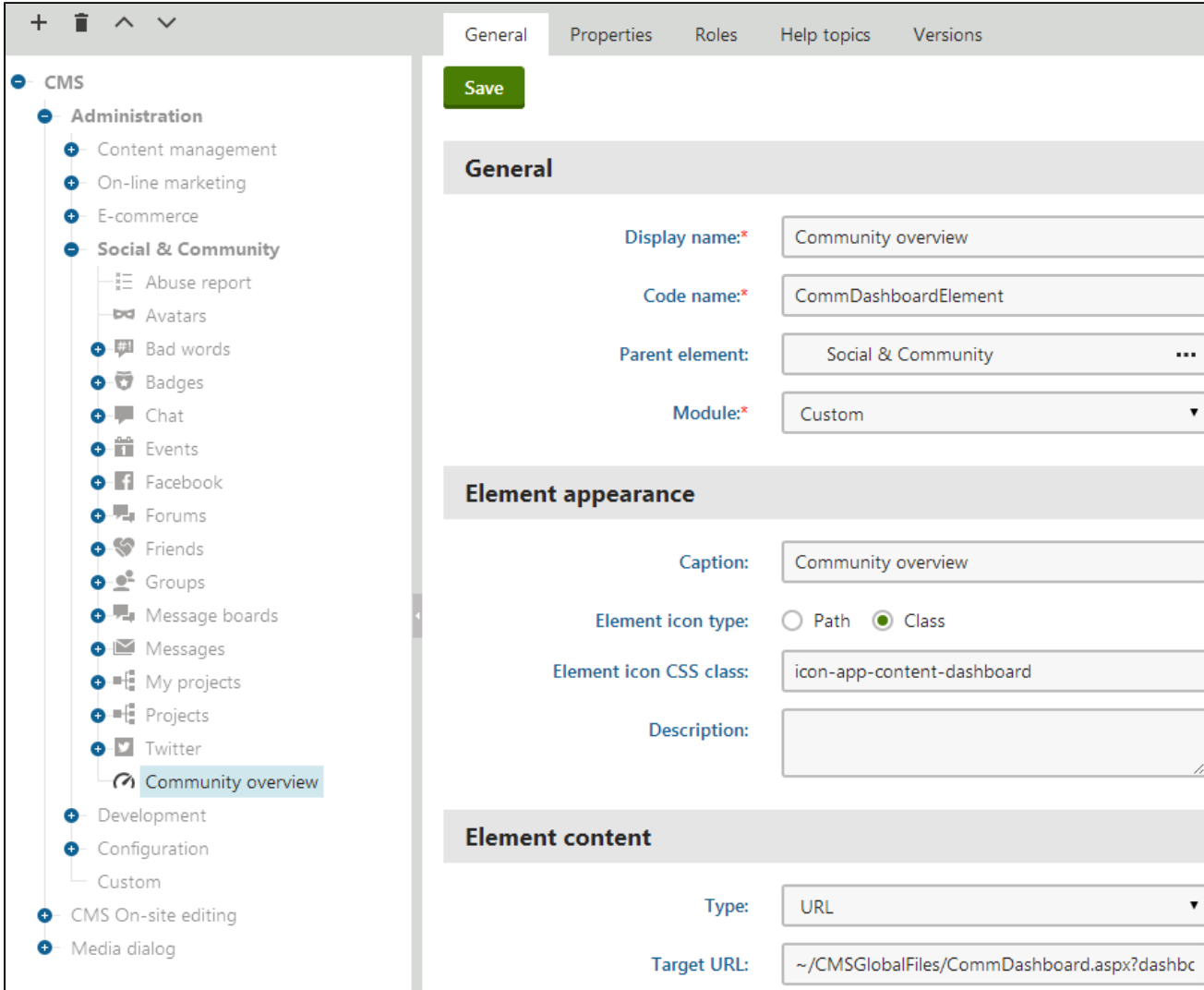
more dashboards share a page template and the *dashboardName* parameters in the URLs used to access the page have the same value, changes made to one of the dashboards also affect the other dashboards (for the given user and site).

- **templateName** - specifies the code name of the page template that the dashboard is based on. The type of the assigned template must be *Dashboard page*. This example uses the template created in the previous steps.
- **hash** - the system automatically generates a hash code for the element

7. Click **Save**.

 When adding applications to the interface of an actual live website, you can set a [macro condition](#) in the **Content permissions** field to define security requirements for access to the application.

The system creates the new UI element.



The screenshot shows the Kentico CMS configuration interface. On the left is a tree view of the CMS structure. The 'Social & Community' section is expanded, and 'Community overview' is selected. The main area shows the configuration for this element, with tabs for General, Properties, Roles, Help topics, and Versions. The 'General' tab is active, showing fields for Display name, Code name, Parent element, and Module. Below this is the 'Element appearance' section with fields for Caption, Element icon type, Element icon CSS class, and Description. The 'Element content' section at the bottom shows the Type (URL) and Target URL.

Field	Value
Display name	Community overview
Code name	CommDashboardElement
Parent element	Social & Community
Module	Custom
Caption	Community overview
Element icon type	Class
Element icon CSS class	icon-app-content-dashboard
Description	
Type	URL
Target URL	~/CMSGlobalFiles/CommDashboard.aspx?dashbc

Creating the dashboard page source file

Now you need to develop the **.aspx** file of the dashboard page in your web project:

1. Open your website in Visual Studio.
2. Create a **New folder** under the root called *CMSGlobalFiles* (if it does not already exist).
3. Rightclick the folder and select **Add -> Add New Item**.
4. Select the **Web Form** template and enable the **Select master page** option.



5. Name the web form *CommDashboard*.

- This is the file specified in the **Target URL** of the previously created UI element. The location ensures that the system exports the file with any site that includes global folders in the export package.

6. Select **~/CMSMasterPages/UI/Dashboard.master** as the master page.

7. Modify the page's markup to match the following:

```
<%@ Page Language="C#" AutoEventWireup="true" MasterPageFile="~/CMSMasterPages/UI/
Dashboard.master" CodeFile="CommDashboard.aspx.cs" EnableEventValidation="false"
Inherits="CMSGlobalFiles_CommDashboard" Theme="Default" %>

<%@ Register Src="~/CMSModules/Widgets/Controls/Dashboard.ascx" TagName="
Dashboard" TagPrefix="cms" %>

<asp:Content runat="server" ID="cplcContent" ContentPlaceHolderID="plcContent">
    <cms:Dashboard ID="ucDashboard" runat="server" ShortID="d" />
</asp:Content>
```



The **Dashboard** user control handles the entire functionality of the dashboard. It processes the query string parameters from the URL used to access the page and displays the corresponding dashboard according to the specified dashboard name, page template and context-related data such as the current site and user.

8. Switch to the web form's code behind.

9. Set the **CMSGlobalFiles_CommDashboard** class to inherit from **DashboardPage**.

10. Modify the class to contain the following code:

```
using System;

using CMS.Core;
using CMS.UIControls;

[UIElement(ModuleName.CUSTOMSYSTEM, "CommDashboardElement")]
public partial class CMSGlobalFiles_CommDashboard : DashboardPage
{
    protected override void OnInit(EventArgs e)
    {
        base.OnInit(e);

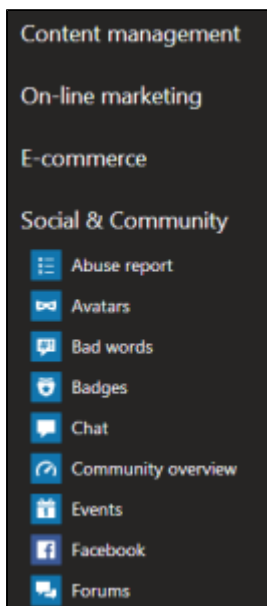
        // Sets up the dashboard and ensures it has unique content for
each site
        ucDashboard.SetupSiteDashboard();
    }

    protected void Page_Load(object sender, EventArgs e)
    {
        // Security access checks for the current user
    }
}
```

11. **Save** both files. If your installation is a web application, **Build** the CMSApp project.

Result

Users can now access the **Community overview** application either through the application list or the application dashboard (if you add the application to the roles of users).



The page displays a fully functional dashboard based on the created page template.

