You can use standard ASP.NET master pages together with ASPX page templates. This is a powerful concept that allows you to share content across all pages without having to add it separately to every page template. For example, you can create master pages containing header and footer sections with a logo, navigation menu, search box etc.

- Define master pages in files with the **.master** extension.
- You can assign one master page to every ASPX page.
- Master pages must always contain one or more **ContentPlaceHolder** controls:

```
<asp:ContentPlaceHolder ID="plcMain" runat="server"></asp:ContentPlaceHolder>
```

The **ContentPlaceHolder** control specifies where child pages display their content inside the master page.

Tip: It is recommended to store master pages in the **CMSTemplates** folder together with your ASPX page template files. This allows the system to export master pages along with your website when you deploy it to another instance of Kentico.

Preparing master pages for ASPX templates

The following code sample shows the markup of a basic master page.

Important

- If you installed the Kentico project as a web application, you need to rename the **CodeFile** attribute on the first line to **Codebehind** for the code example to be functional.
- The **CodeFile/Codebehind** attribute's value must match the name of the master page's code behind file.
- Set the value of the *Inherits* attribute according to the location and name of the master page file.

```
<%@ Master Language="C#" AutoEventWireup="true" CodeFile="Custom.master.cs" Inherits="
CMSTemplates_CorporateSite_Custom" %>

<%=DocType%>

<html xmlns="http://www.w3.org/1999/xhtml" <%=XmlNamespace%>>
<head id="Head1" runat="server">
        <title id="Title1" runat="server">My site</title>
        <asp:literal runat="server" id="ltlTags" enableviewstate="false" />
</head>

<body class="<%=BodyClass%>" <%=BodyParameters%>>
        <form id="form1" runat="server">
                <asp:PlaceHolder runat="server" ID="plcManagers">
                        <asp:ScriptManager ID="manScript" runat="server" ScriptMode="
Release" EnableViewState="false" />
                        <cms:CMSPortalManager ID="CMSPortalManager1" runat="server"
EnableViewState="false" />
                </asp:PlaceHolder>

                <cms:CMSListMenu ID="CMSListMenu1" runat="server" />

                <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
                </asp:ContentPlaceHolder>
        </form>
</body>
</html>
```

All ASPX page templates require the following manager controls, so it is a good practice to add them onto your website's master page:

Control name	Description
asp:ScriptManager	Allows pages to use AJAX components. If required, the **CMSPortalManager** automatically loads the *ScriptManager*, but adding the control directly reduces overhead.
CMSPortalManager	Ensures the transferring of content between the database and editable regions. Also provides the management functionality needed for portal engine zones. • The CMSPortalManager must be placed inside a standard PlaceHolder control. **Note**: If your master page contains the **CMSPageManager** control, remove it and add the **CMSPortalManager** instead. The *CMSPageManager* is an older equivalent, which is obsolete and only available for the purposes of backwards compatibility.

The CMSListMenu control is one of the options that you can use to generate a menu for website navigation.

Writing the master page code behind

You need to modify the code behind file of your master pages according to the following steps:

1. Add a reference to the **CMS.UIControls** namespace:

```
using CMS.UIControls;
```

2. Change the class definition to match the following (the name of the class may be different):

```
public partial class CMSTemplates_CorporateSite_Custom : TemplateMasterPage
```

Master pages of ASPX templates must always inherit from the **TemplateMasterPage** class.

3. Add the following code into the master page's code behind class:

Adjust the value of the **PageManager** property according to the ID of the *CMSPortalManager* control placed on the master page.

```
protected override void CreateChildControls()
{
    base.CreateChildControls();
    PageManager = CMSPortalManager1;
}

protected override void OnPreRender(EventArgs e)
{
    base.OnPreRender(e);
    this.ltlTags.Text = this.HeaderTags;
}
```

This code ensures that ASPX templates using the given master page support all required functionality.

Now, you can use the new master page to create ASPX page templates.