This page is a part of a tutorial, which you should follow sequentially, from the beginning to the end. Go to the first page: Getting started with Kentico.

In the previous step of this tutorial, you have implemented the functionality that retrieves page data from the database and displays it on the live site. In this step, you will code a dynamic menu that allows your content editors to manage the website's navigation.

You will learn about:

- Creating a view model for the menu items
- Creating the Menu controller
- Creating the navigation menu view
- Updating the website layout

Website navigation tends to change based on the site's current content and the requirements of its owner. The *MenuItem* page type allows website editors to select which pages from the content tree appear in the navigation menu. The menu's presentation, design and behavior are then handled in the MVC application. We recommend using the strongly-typed classes that you generated for the *M enuItem* page type to retrieve the menu data.

Creating a view model for the menu items

1. In Visual Studio, create a new **Menu** subfolder in the **Models** folder.
2. Select the *Menu* subfolder and add a new **MenuItemViewModel** class.
3. Define the view model properties using the following code:

```
namespace MedioMVC.Models.Menu
{
    public class MenuItemViewModel
    {
        // Defines the properties of the MenuItem view model
        public string MenuItemText { get; set; }
        public string MenuItemRelativeUrl { get; set; }
    }
}
```

4. Save your changes.

Creating the Menu controller

The controller class will define a *GetMenu* action that loads and displays the content of the site's navigation menu. The action retrieves the menu data using the generated *MenuItemProvider* class, then loads all pages which content editors selected as menu items, and organizes the data into a collection of models that gets passed to the view.

1. In the **Controllers** folder, create a new **MenuController**class.
2. Replace the default code with the following:

```
using System.Linq;
using System.Web.Mvc;

using CMS.DocumentEngine;
using CMS.DocumentEngine.Types.MEDIO;

using MedioMVC.Models.Menu;

namespace MedioMVC.Controllers
{
    public class MenuController : Controller
    {
        // GET: Loads and displays the site's navigation menu
        public ActionResult GetMenu()
        {
            // Loads all menu items using the page type's generated provider
            // Uses the menu item order from the content tree in the Kentico
'Pages' application
            var menuItems = MenuItemProvider.GetMenuItems()
                .Columns("MenuItemText", "MenuItemPage")
                .OrderBy("NodeOrder");

            // Loads the pages selected within the menu items
            // The data only contains values of the NodeGUID identifier column
            var pages = DocumentHelper.GetDocuments()
                .WhereIn("NodeGUID", menuItems.Select(item => item.MenuItemPage).
ToList())
                .Columns("NodeGUID");

            // Creates a collection of view models based on the menu item and page
data
            var model = menuItems.Select(item => new MenuItemViewModel()
            {
                MenuItemText = item.MenuItemText,
                // Gets the URL for the page whose GUID matches the given menu
item's selected page
                MenuItemRelativeUrl = pages.FirstOrDefault(page => page.NodeGUID ==
item.MenuItemPage).RelativeURL
            });

            return PartialView("_SiteMenu", model);
        }
    }
}
```

3. Save your changes.

The methods of the generated page type providers, as well as the more general *DocumentHelper.GetDocuments* method, work using the Kentico **DocumentQuery** API. DocumentQuery provides an abstraction layer over your SQL database, and allows you to call additional methods that adjust how the page data is retrieved.

For example, the *OrderBy* method sets the order of the retrieved data items, and the *Columns* method ensures that the underlying SQL query only loads the data columns that you need. For performance reasons, we strongly recommend limiting columns in all data retrieval API calls.

See the documentation if you wish to learn more: Working with pages in the API

Creating the navigation menu view

A navigation menu is an element that appears on most pages of a website. For this reason, you will define the menu's output in a partial view, which can then be added into the site's main layout (or into the views of other pages if necessary).

1. In the *MenuController*, right-click the **PartialView** method and select **Add View**.
2. Fill in the following values:
   - **View name**: _SiteMenu
   - **Template**: Empty
   - **Create as a partial view**: Yes (selected)
3. In the view code, add a using statement for the *MedioMVC.Model.Menu* namespace.
4. Add the *@model* directive and specify the model type as an *IEnumerable* collection of *MenuItemViewModel*.
5. Define the menu output by rendering HTML links within a *<nav>* element (using the menu item URL and text values from the model). The partial view code should look like this:

```
@using MedioMVC.Models.Menu;
@model IEnumerable<MenuItemViewModel>

<nav>
    @foreach (MenuItemViewModel menuItem in Model)
    {
                @* Iterates through the view model collection and renders HTML
links for the menu items *@
        <a href="@Url.Content(menuItem.MenuItemRelativeUrl)" title="@menuItem.
MenuItemText">@menuItem.MenuItemText</a>
    }
</nav>
```

6. Save your changes.

Updating the website layout

You have coded the functionality behind the navigation menu. Let's put the code to use and add the menu to the site's layout.

1. Edit your *_Layout.cshtml* file (in the *Views/Shared* folder).
2. Locate the section defined by *<nav>* tags, and replace it with the **Html.Action** Razor call. Invoke the *GetMenu* action in the *Me nu* controller.
3. Locate the *<a>* element in the *<header>* section that displays the *MEDIO clinic* logo, and set the value of the *href* tag to: ~/

The tilde character with the forward slash (~/) ensures that the Medio clinic logo link targets the root page of the website.

Your final *_Layout.cshtml* markup should look like this:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    @* Dynamically resolves the page's title *@
    <title>Medio Clinic - @ViewBag.Title</title>

    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
1q8mTJOASx8j1Au+a5WDVnPi2lkFfwwEAa8hDDdjZlpLegxhjVME1fgjWPGmkzs7" crossorigin="anonymous">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.6.3
/css/font-awesome.min.css">
    <link href="http://fonts.googleapis.com/css?family=Lato:400,700italic&amp;
```

```
subset=latin,latin-ext" rel="stylesheet" type="text/css">
    <link href="http://fonts.googleapis.com/css?family=Lobster" rel="stylesheet" type="
text/css">

    @* Loads the style.css resource *@
    <link rel="stylesheet" href="~/Content/Styles/styles.css">
</head>
<body>
    <header>
        <div class="col-sm-offset-3 col-sm-6">
            <div class="col-sm-6">

                                @* Targets the root page of the website *@
                <a href="~/" title="MedioClinic homepage" class="logo">MEDIO clinic</a>
            </div>
            <div class="col-sm-6 nav">
                                @* Loads the partial view with the navigation menu,
including the <nav> element *@
                @Html.Action("GetMenu", "Menu")
            </div>
        </div>
        <div class="clearfix"></div>
    </header>

    @* Loads the content of your Tutorial's pages as sub views *@
    @RenderBody()

    <footer>
        <div class="col-sm-offset-3 col-sm-6">
            <div class="row">
                <div class="col-sm-6">
                    <h4>MEDIO clinic</h4>
                    <ul>
                        <li><i class="fa fa-map-marker"></i> Address: <address>7A Kentico
street, Bedford, NH 03110, USA</address></li>
                        <li><i class="fa fa-envelope-o"></i> E-mail: <a href="mailto:
info@medio-clinic.com" title="Email us">info@medio-clinic.com</a></li>
                        <li><i class="fa fa-phone"></i> Phone number: <a href="tel:
5417543010" title="Phone us">(541) 754-3010</a>
                    </ul>
                </div>
                <div class="col-sm-6">
                    <span class="cms">Powered by <a href="http://www.kentico.com" title="
Kentico CMS">Kentico CMS for ASP.NET</a></span>
                </div>
            </div>
        </div>
        <div class="clearfix"></div>
    </footer>
</body>
</html>
```

Build your project and test your navigation menu on the live site website. Because the original layout already contained hard-coded HTML links, you won't see a difference on the live site. However, you can test the menu's dynamic functionality by adding another menu item in the Kentico administration interface (*Pages* application), or by changing the order of the *Home* and *Medical center* items. The implementation you've built ensures any menu item changes are immediately displayed in the navigation menu.

You can download packages with the **exported MEDIO clinic MVC website** and **MedioMVC project files**. To compare the sample solution with your own implementation, import the website into your Kentico installation in the **Sites** application or view the code of the MVC application in Visual Studio.

Previous page: Displaying the content of the pages — **Next page:** Next steps in MVC development

Completed pages: 9 of 10