**Aarhus Web Developers Network Documentation**

**By Marian Zoicas**

Table Of Content

[1. Introduction 3](#_Toc509832492)

[2. Sitemap 3](#_Toc509832493)

[3. The website menu 3](#_Toc509832494)

[4. Website access control 4](#_Toc509832495)

[5. Contact Form 7](#_Toc509832496)

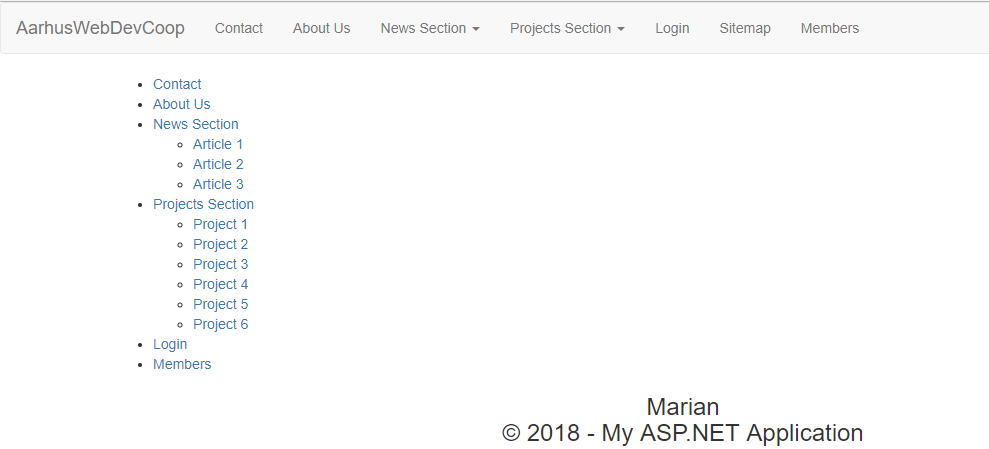
[6. Conclusion 8](#_Toc509832497)

## Introduction

On the first part of the Development Environments class, we had the first mandatory project, which is a web solution of Aarhus Web Developers Network. The task was made with Visual Studio and with Umbraco CMS. I included in the project seven different pages, starting with Contact page, where I created a Contact form, an About Us page made with Umbraco’s Grid Layout, A news page with articles, a Projects page with completed, initiated, idea and under development projects, witch are visible, depending on the roles that the user has. The last three pages are: Login, Sitemap and the Members page.

## Sitemap

I created the Sitemap in order to present all the pages and the subpages of the website.



## The website menu

After I made the pages, the next step was to include a navigation bar for the website. I created a Partial View with navigation snippet, and Umbraco auto-generate the code for me. In order to make the dropdown for the subpages included in the menu, I added extra code, more exactly, it is a condition, if the page had more than one children, it will create a dropwdown list. I am calling the Main Menu partial view on the Master template using the

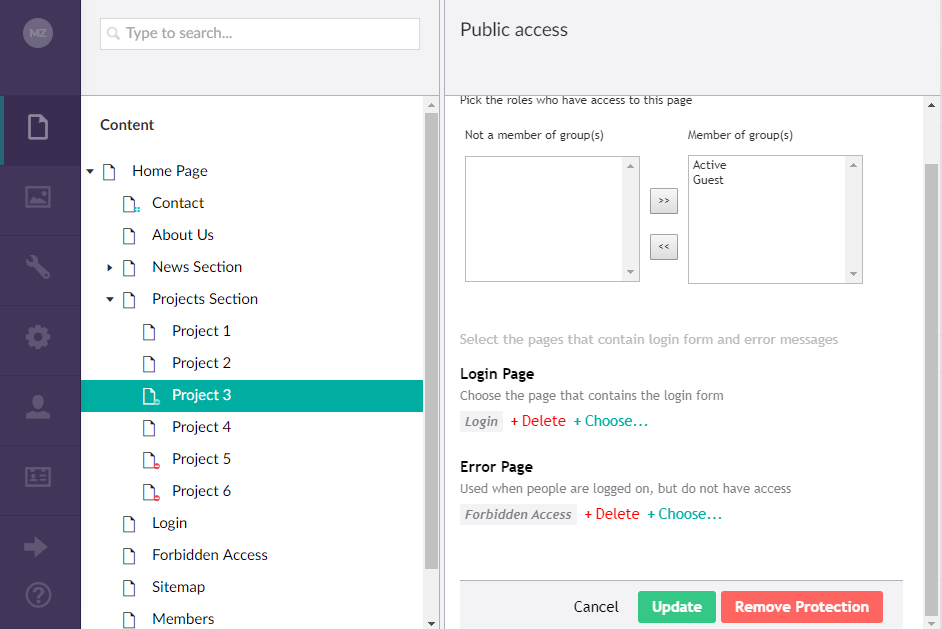
@Umbraco.RenderMacro("mainMenu").



## Website access control

Some parts of the website had to be protected, for that we created a login from snippet partial view to restrict the access for some of the users. I included an if is logged in statement, where will display the form, and an else condition where will show the status” You are currently logged in”. I created the log out button in order to give the users the opportunity to log out.

In order to use the role based protection, I also created a Forbiden Access page with NaviHide property set to default as true, which will be displayed in case the users won’t have acces to the specific page. I used the public access option in the content tree for the project number 3 for example, which is an initiated project, which has to be visible for active and guests members, not for public.



I added two conditions in the Nav Menu partial view file, on the pages that has selected the protection from the public access content tree. One for the menu items, and one from the submenu items, and if the users have access after the login, then it will show the page for them.



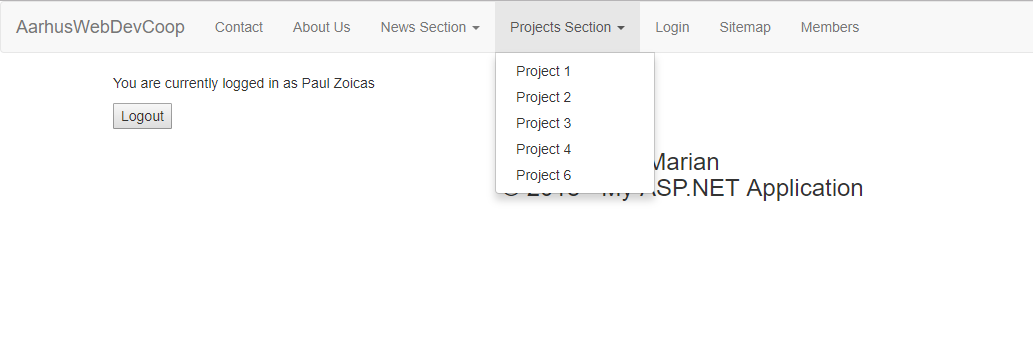
On the next screenshots, I am going to show the access for the completed, idea, under development and initiated projects based on public, active or guests views.

The completed projects are Projects number 1, 2,4. Project number 3 is initiated, project 5 is idea and 6 is under development.

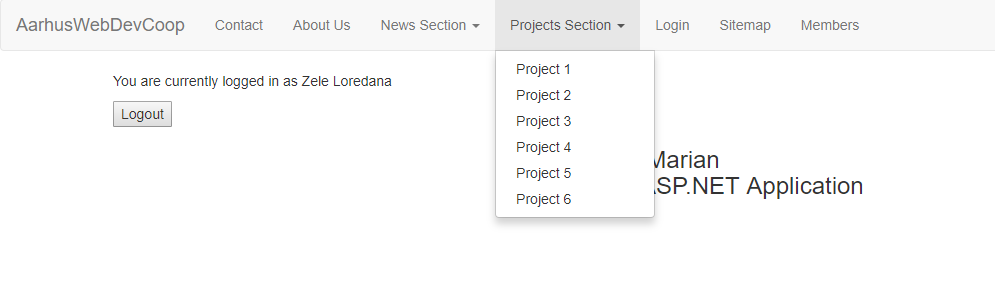
The first image represents the public view.



The second view represents the guests view.



The third view is the active view with full access on projects.



## Contact Form

For the Contact page, I created in Visual Studio a view model with 4 classes, and then I created the view based on the view model. I created a controller with the instance of the View Model class and that returns the contact form view. The controller includes two models, the Get and Post methods. For handling the posts, I used settings to receive the posts as in email format. I declared a method that receives the form input and I created a message object inside the method which will receive the data of type Contact Form.

The following step is for receiving the post as email, a problem that I fixed it by adding Gmail credentials, so in case a post is submitted, I will receive the post on my gmail account and it will be display also in the content tree on the Contact Us section.



I added the validations in the view model, and I am calling them on the Contact Us view page, using the @Html.ValidationMessageFor.

In order to include the Client Side Validation, I installed in the Nugget Console:

Package jQuery.Validation

Package Microsoft.jQuery.Unobtrusive.Validation

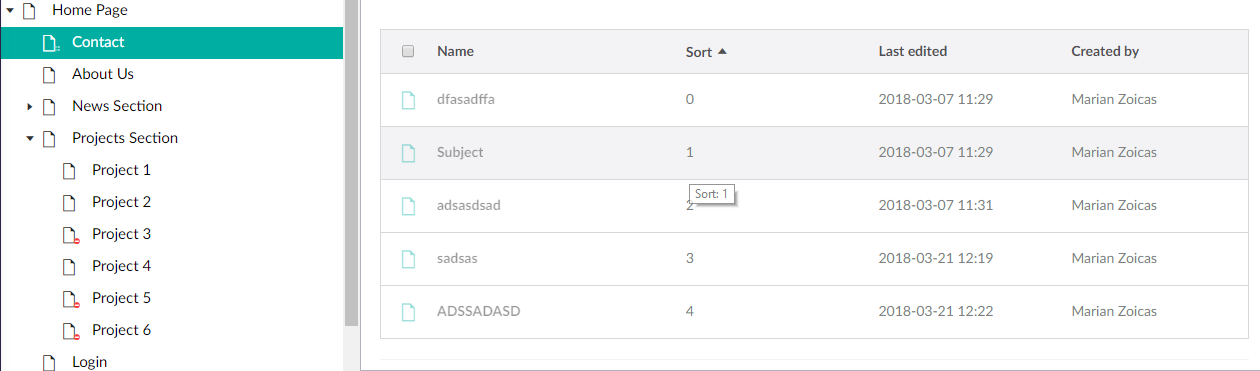
I enable the Client Side validation from the web config files by adding two child nodes in the appsettings:

<Add key="ClientValidationEnabled" value="true"/>

<addkey="UnobtrusiveJavaScriptEnabled" value="true"/>

On the next step, I created a variable called TempData in Controller, and I included in the partial view, so in case the form has no errors and it is sent, it will give a feedback that includes a message: “Thank you for contacting us. We’ll be in touch soon!”

On the final part, I created a new message document type, with Name, Email, Subject and Message Content properties, and I created a Created content using the Content API with the same properties names as in the document type in order to receive the messages in the Contact form section in the content tree.



## Conclusion

In conclusion, I can affirm that it was a really interesting project, having the possibility to work with both Visual Studio and Umbraco, helped me to learn more things about Visual Studio and also learning the functionalities and the features of the Umbraco CMS. I had a lot of challenges on finishing the project, but with a lot of hours of practicing with Umbraco, I can proudly say that I gained important knowledge and I can proudly add the project to my Portfolio.