FREE eBook

LEARNING umbraco

Free unaffiliated eBook created from
Stack Overflow contributors.

#umbraco

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: umbraco

It is an unofficial and free umbraco ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official umbraco.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with umbraco

Remarks

Umbraco is an open source, MIT-Licensed .NET content management system. Initially created by Danish developer Niels Hartvig in 2000 as a hobby project, Umbraco was released as open source in 2004 and has since been developed and maintained continuously by a core team made up of paid Umbraco employees and community members.

Developers using Umbraco can benefit from it's mature codebase, high degree of customisability, and large, friendly community. Contrary to other CMS systems, Umbraco does not constrain a developer to a "theme" or "plugin" based workflow. Umbraco instead provides a rich API on top of ASP.NET MVC, making existing .NET web developers feel at home when writing custom Umbraco solutions.

Editors using Umbraco can benefit from the intuitive "backoffice", a mobile-responsive AngularJS app which makes content-writing and admin tasks easy, and from the large amount of editor-focused documentation available including umbraco.tv.

Versions

Version	Comments	Release date
4.11.10	Version 4 was initially released in 2010. Is still in use, however not actively developed.	2013-06-27
5	Version 5 was released in 2011, but had major performance and code complexity problems, and this version was dropped.	2011-09-05
6.2.6	Version 6 was released in 2013, and focused on MVC 4 and API	2016-03-31
7.5.3	Version 7 was released late in 2013. It is updated concurrently with version 6, and the major new features are a redesign of the back-office user interface	2016-09-05
7.6.2	Template experience update, added new colors, script and template editors	2017-05-30

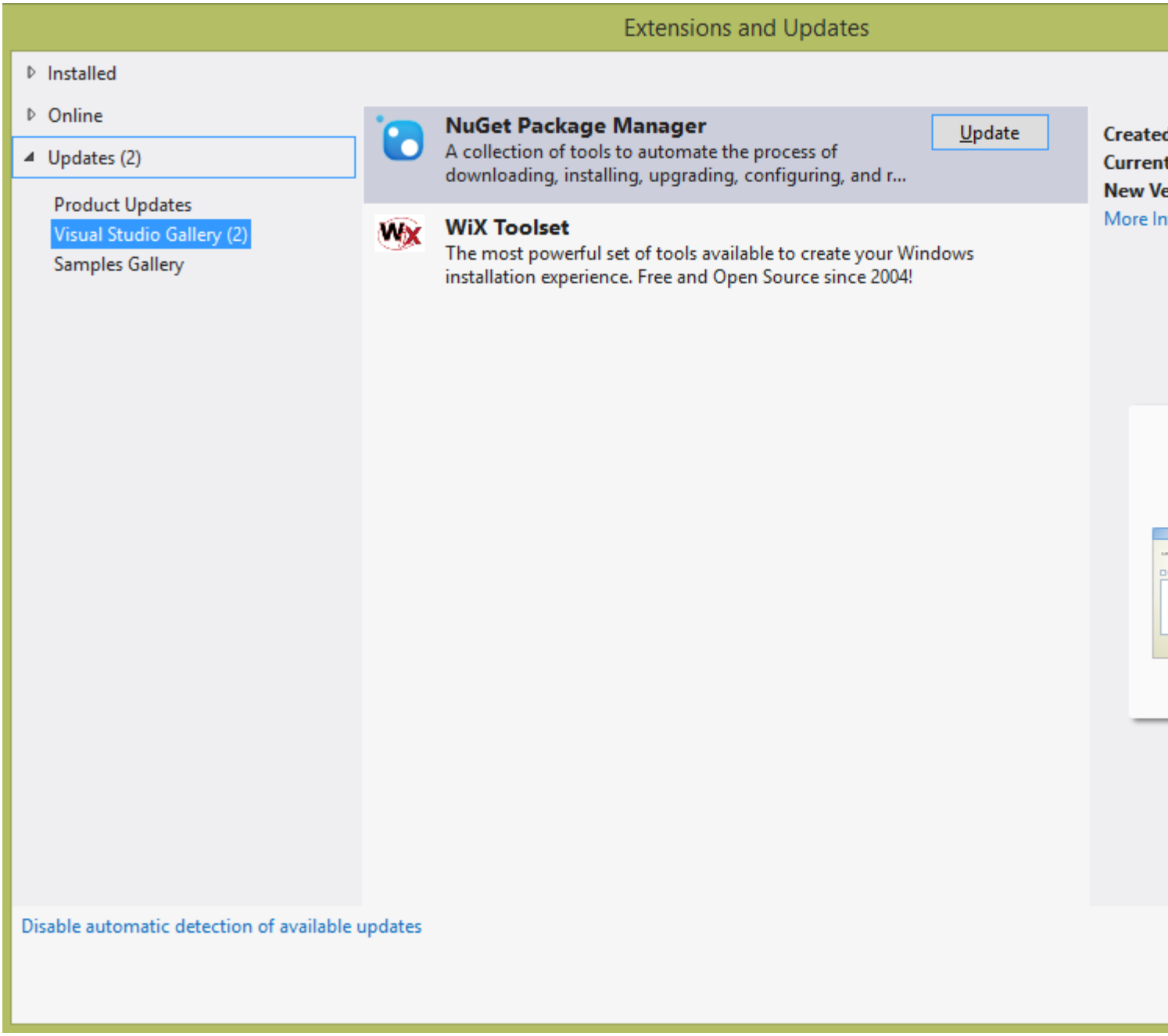
Examples

Install Umbraco with NuGet

NuGet version

Before you start: make sure your NuGet version is up to date.

In Visual Studio, go to Tools > Extensions and Updates, then Updates > Visual Studio Gallery. Check if there is a NuGet Update available and install it. Or, you can uninstall the existing nuget and reinstall it. It will install the latest version of nuget.



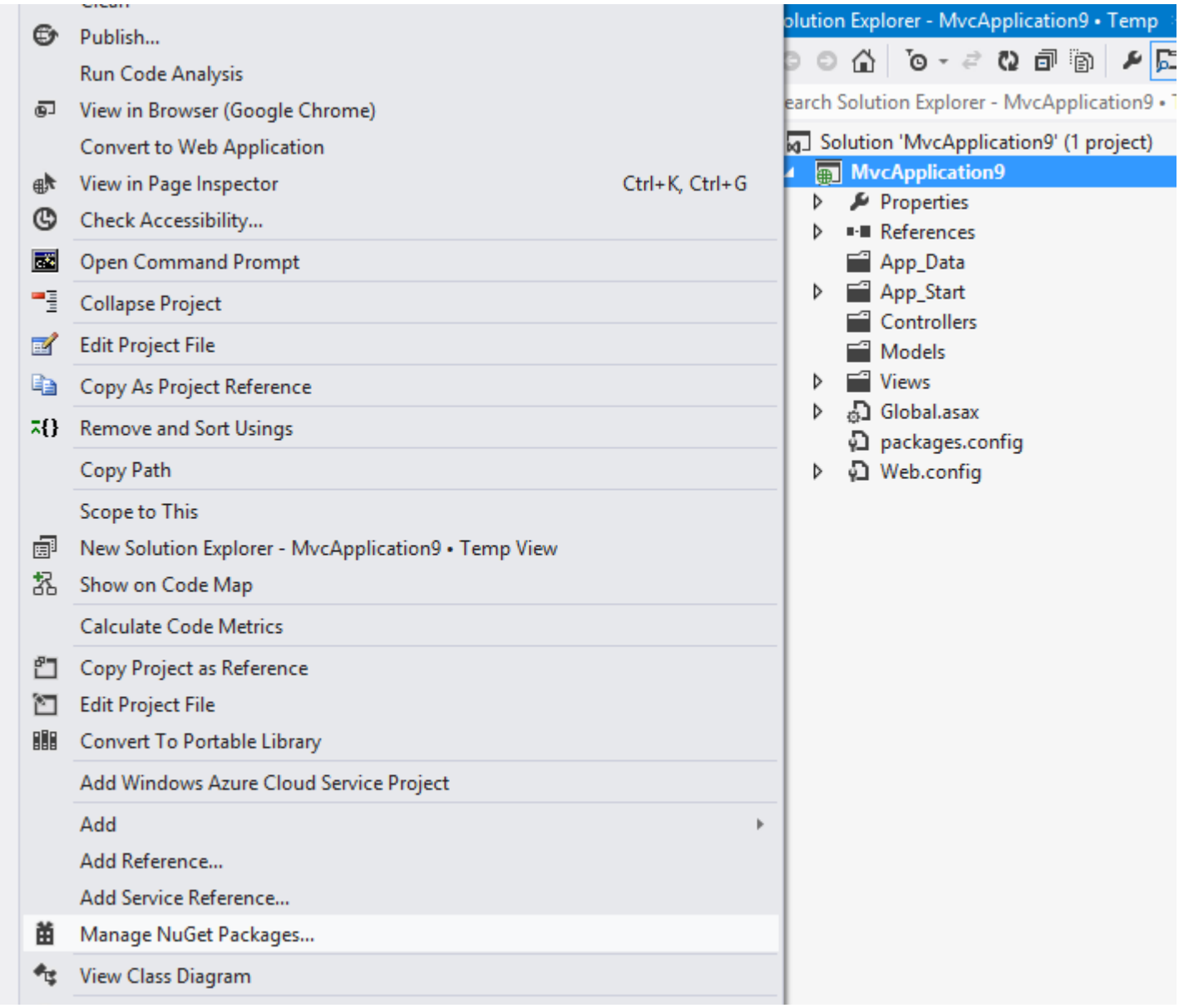
.Net Version

If you're installing Umbraco 7+ then you need to choose .NET Framework 4.5 or 4.5.1 here. For Umbraco 6 you can still choose .NET Framework 4, but 4.5 and 4.5.1 also work.

You can select the version from the create new project window of visualstudio.

Project:

Create an empty asp.net web application. Then, right-click on the new project you just made and choose Manage NuGet Packages.



Find the "UmbracoCms" package, and click install. NuGet will then download dependencies and will install all of Umbraco's files in your new solution.

It will ask you to overwrite your web.config file. Confirm with yes. Build and Run solution (CTRL-SHIFT-B).

After the build a web browser will open, and you can continue installation in the web interface. You have to configure the database and select a template (if you want any template to start with).

> Above information are from umbraco official site.for more details go to this link

Read Getting started with umbraco online: https://riptutorial.com/umbraco/topic/1768/getting-started-with-umbraco

Chapter 2: Installation of Umbraco CMS

Examples

Hosting enviroment

For Umbraco 7 the requirements are

- IIS 7 or higher
- Database, one of the following: SQL CE, SQL Server 2008 or higher or MySQL with support for case insensitive queries)
- ASP.NET 4.5 or 4.5.1. Full-Trust
- Ability to set file/folder permissions for the user that "owns" the Application Pool

Manual installation of Umbraco

1. Download the files from our.umbraco.org/download and unzip them in a folder.
2. Set up web server hosting the folder you chose. Although IIS is a requirement for production sites, it runs fine for development in IIS Express.
3. Set up file permissions for the folder. For development server it is okey to have full read/write/modify permissions for the process serving the web page (AppPool or Network service)
4. If you want to set up Umbraco on Sql Server or Mysql, download and install it. Make a new database for Umbraco, and remember Connectionstring details, inccluding database name, user and password
5. Open your site at localhost, and follow the wizard to set up your site with the database.

Install Umbraco with NuGet

1. Check for updates to the Nuget package manager. In Visual Studio: Tools > Extensions and Updates > Updates > Visual Studio Gallery. Install if availalbe
2. Create a new web application with template "ASP.NET Web Application with an Empty template" on .NET Framework 4.5.1
3. Open the package manager console and run `Install-Package UmbracoCms`
4. Press F5 to build and run your new website.
5. Complete the wizard to choose database provider and set up your site.

Use Umbraco as a cloud service

Either you just want to test out Umbraco CMS, or host your site in a cloud service, you could sign up for a free trial at umbraco.com/cloud. The site you develop in the cloud service could be downloaded for local development or your own hosting later.

Read Installation of Umbraco CMS online: https://riptutorial.com/umbraco/topic/7325/installation-of-umbraco-cms

Chapter 3: Make a SurfaceController to allow members to log in

Remarks

The code was written and tested on Umbraco 7.5.3.

Examples

Add ViewModel

First we need to make a ViewModel. Make a new folder /Models/Account, and add the file MemberLoginViewModel.cs to the folder.

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace MyCMS.Models.Account
{
    public class MemberLoginViewModel
    {
        [Required]
        [EmailAddress]
        [Display(Name = "Email")]
        public string Email { get; set; }

        [Required]
        [DataType(DataType.Password)]
        [Display(Name = "Password")]
        public string Password { get; set; }

        [Display(Name = "Remember me?")]
        public bool RememberMe { get; set; }
    }
}
```

Add a view

Then make a view in the folder /Views/MacroPartials/Account

```
@model MyCMS.Models.Account.MemberLoginViewModel
@using MyCMS.Controllers.Account

@if (User.Identity.IsAuthenticated)
{
        <p>Logged in as: @User.Identity.Name</p>
        using (Html.BeginUmbracoForm<MemberLoginSurfaceController>("MemberLogout",
```

```
FormMethod.Post, new { id = "logoutForm" }))
        {
            @Html.AntiForgeryToken()
         <button type="submit" class="btn btn-outline-secondary">Log out</button>
         }

}
else
{
    using (Html.BeginUmbracoForm<MemberLoginSurfaceController>("MemberLoginPost",
"MemberLoginSurface", new { @class = "form-inline" }))
    {
        <div class="form-group">
            <label for="email">Email</label>
            <input name="Email" type="email" class="form-control" id="loginform_email"
placeholder="name@email.no">
        </div>
                <div class="form-group">
                    <label for="password">Password</label>
                    <input name="Password" type="password" class="form-control"
id="loginform_password" placeholder="*****">
                </div>
                <div class="form-check">
                    <label class="form-check-label">
                        <input name="RememberMe" class="form-check-input" type="checkbox">Husk
meg
                    </label>
                </div>
                <button type="submit" class="btn btn-outline-secondary">Log in</button>

    }
    <p>@TempData["Status"]</p>
}
```

Add controller

Make a new controller in the folder /Controllers/Account. Name the file MemberLoginSurfaceController.cs

```
using MyCMS.Models.Account;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Security;

namespace MyCMS.Controllers.Account
{
    public class MemberLoginSurfaceController : Umbraco.Web.Mvc.SurfaceController
    {
        // Inspired by: http://24days.in/umbraco/2012/creating-a-login-form-with-umbraco-mvc-
surfacecontroller/

        // GET: MemberLoginSurface
        [HttpGet]
        [ActionName("MemberLoginForm")]
        [ChildActionOnly]
        public ActionResult MemberLoginForm()
```

```
        {
            return PartialView("Components/MemberLogin", new MemberLoginViewModel());
        }

        // The MemberLogout Action signs out the user and redirects to the site home page:
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult MemberLogout()
        {
            TempData["Message"] = "You have logged out!";
            Session.Clear();
            FormsAuthentication.SignOut();
            return RedirectToCurrentUmbracoPage();
        }

        // The MemberLoginPost Action checks the entered credentials using the standard Asp
Net membership provider and redirects the user to the same page. Either as logged in, or with
a message set in the TempData dictionary:

        [HttpPost]
        [ActionName("MemberLoginPost")]
        public ActionResult MemberLoginPost(MemberLoginViewModel model)
        {

            if (Membership.ValidateUser(model.Email, model.Password))
            {
                FormsAuthentication.SetAuthCookie(model.Email, model.RememberMe);
                return RedirectToCurrentUmbracoPage();
            }
            else
            {
                TempData["Status"] = "Invalid username or password";
                return RedirectToCurrentUmbracoPage();
            }
        }
    }
}
```

Show me the login form!

To render your new component in a view, you cust call:

```
@Html.Action("MemberLoginForm", "MemberLoginSurface")
```

Read Make a SurfaceController to allow members to log in online:
https://riptutorial.com/umbraco/topic/7318/make-a-surfacecontroller-to-allow-members-to-log-in

Credits

S. No	Chapters	Contributors
1	Getting started with umbraco	Community, Cynthia Tiwana, glcheetham, MCollard, Ole Kristian Losvik, Sadid Khan
2	Installation of Umbraco CMS	Ole Kristian Losvik
3	Make a SurfaceController to allow members to log in	Ole Kristian Losvik