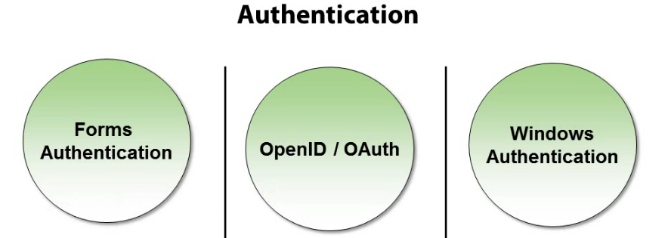
Използване на session обект за запазване на login информацията и ауторизация.

На теория не е препоръчително да се пази session информацията по този начин, но на практика, това е най-лесния и удобен начин

Има 3 стандартни начина за ауторизация. Ние ще покажем наш, опростен вариант, подобен на Forms

Authentication .



------------------

Създаваме нов празен MVC проект FormLogin

------------------

Създаваме празен HomeController

-------------------

Добавяме  Index view и в него линк към Login

@Html.ActionLink("Login", "Login", "Home")

-------------------

Създаваме viewModel class LoginViewModel с 2 полета

namespace FormLogin.Models

{

public class LoginViewModel

{

[Required]

public string Username { get; set; }

[Required]

[DataType(DataType.Password)]

public string Password { get; set; }

}

}

=================

Създаваме GET action Login() в HomeController

[HttpGet]

public ActionResult Login()

{

return View();

}

----------------------

Създаваме и view Login.cshtml с Template: “Edit” и Model class: LoginViewModel (FormLogin.Models)

T.е. то ще е строго типизирано.

-----------------------

Може временно да закоментираме реда @Html.AntiforgeryToken() за да може да пробваме с произволни входни данни.

-------------------------

Създаваме POST action LoginPost(LoginViewModel viewModel)  и му слагаме атрибут [ActionName("Login")]

[HttpPost]

[ActionName("Login")]

public ActionResult LoginPost(LoginViewModel viewModel)

{

if (!ModelState.IsValid)

{

return View();

}

else

{

return RedirectToAction("Index");

}

}

---------------------------------

Вече работи server-side валидацията

Добавяме “ jquery validation” и “Microsoft.jQuery.Unobtrusive.validation”

Вече и client-side валидацията трябва да работи при login

------------------------

Добавяме нова директория Helpers в проекта и в нея class LoginUserSession.cs

Добавяме property-та само най-важните неща които искаме да се пазят в сесията - засега са UserID, Username, IsAuthenticated, IsAdministrator.

IsAuthenticated е най важното поле което ще пази информация дали имаме валидна сесия или не

Написваме “public static LoginUserSession Current {get{....}}”

Целта е да работим с една инстанция на обекта за сесията, както и да сме сигурни, че обекта никога няма да е null. Реално най-често ще проверяваме:

LoginUserSession.Current.IsAuthenticated

А другите полета ще ги вземаме със LoginUserSession.Current.XXX

public class LoginUserSession

{

public int UserID { get; private set; }

public string Username { get; private set; }

public bool IsAuthenticated { get; private set; }

public bool IsAdministrator { get; private set; }

private LoginUserSession()

{ // IsAuthenticated = false;

}

public static LoginUserSession Current

{

get

{

// get the LoginUserSession object in the session and if it is null -> create new object and set it in the session

LoginUserSession loginUserSession = (LoginUserSession)HttpContext.Current.Session["LoginUser"];

if (loginUserSession == null)

{

loginUserSession = new LoginUserSession();

HttpContext.Current.Session["LoginUser"] = loginUserSession;

}

return loginUserSession;

}

}

public void SetCurrentUser(int userID, string username, bool isAdministrator)

{

this.IsAuthenticated = true;

this.IsAdministrator = isAdministrator;

this.UserID = userID;

this.Username = username;

}

public void Logout()

{

// set the property values to default values

// and the most important one is IsAuthenticated = false

this.IsAuthenticated = false;

this.IsAdministrator = false;

this.UserID = 0;

this.Username = string.Empty;

}

}

------------------------------------

Променяме Home/Index.cshtml и добавяме проверката дали сесията е валидна и добавяме линк за Logout, ако е валидна - @if(LoginUserSession.Current.IsAuthenticated)...

@using FormLogin.Helpers

@{

ViewBag.Title = "Index";

}

<h2>Index</h2>

@if (LoginUserSession.Current.IsAuthenticated)

{

<div>Hello @LoginUserSession.Current.Username </div>

<div>You are administrator: @LoginUserSession.Current.IsAdministrator </div>

@Html.ActionLink("Logout", "Logout", "Home")

}

else

{

@Html.ActionLink("Login", "Login", "Home")

}

==============================

Добавяме mockup classes за User и UserRepository, които симулират реални обекти в тестови приложения

namespace FormLogin.Controllers

{

#region mockup classes

// This code is only for testing

// In real application, you should have your own classes User and UserRepository

public class User

{

public int ID { get; set; }

public string Username { get; set; }

public bool IsAdministrator { get; set; }

}

public class UserRepository

{

public User GetUserByNameAndPassword(string username, string password)

{

return new User() { ID = 2, Username = username, IsAdministrator = true };

}

}

#endregion

public class HomeController : Controller

{

. . . . .

}

}

=============================

Променяме LoginPost() за да направим проверка дали съществува User в “базата” с подадените username и password от view-то. Ако съществува - set-ваме го в сесията и пренасочваме към Index. Ако не същестува, показваме validation error.

Добавяме public ActionResult Logout()

[HttpPost]

[ActionName("Login")]

public ActionResult LoginPost(LoginViewModel viewModel)

{

if (ModelState.IsValid)

{

// here we have to check if the username exists in the database

UserRepository userRepository = new UserRepository();

User dbUser = userRepository.GetUserByNameAndPassword(viewModel.Username, viewModel.Password);

bool isUserExists = dbUser != null;

if (isUserExists)

{

LoginUserSession.Current.SetCurrentUser(dbUser.ID, dbUser.Username, dbUser.IsAdministrator);

return RedirectToAction("Index");

}

else

{

ModelState.AddModelError("", "Invalid username and/or password");

}

}

// if we are here, this means there is some validation error and we have to show the login screen again

return View();

}

public ActionResult Logout()

{

LoginUserSession.Current.Logout();

return RedirectToAction("Index");

}

=============================================

Тестваме че всичко работи.