ASP.NET WEB PAGES – MEMBERSHIP

thecodingguys

11 January 2013

This document covers how to add additional functionality to an ASP.NET registration system, I will go over a few things and a 2 step-verification process using text local SMS gateway.

CONTENTS

ABOUT

This document is not a tutorial on how to add registration to an ASP.NET website it continues on from the tutorial at http://www.thecodingguys.net/tutorials/asp/webpages-membership. I am assuming you have already followed that and you want to implement features not discussed there such as additional user information, registration via username, account locked out etc.

I will go over a few things that I did not cover there, considering the membership tutorials are popular I decided to make this guide for the users who are lost and want extra functionality I will talk about:

- Adding additional user information (such as name, number etc)
- Registration via username
- Account locked out status
- 2 Step-Verification using textlocal SMS gateway (Yes it's possible!)

LICENSE

ADDING ADDITIONAL USER INFORMATION

Adding extra user information (such as their age, gender etc) is quite easy, when we registered the user as seen in this code:

```
try{

 var token = WebSecurity.CreateUserAndAccount(email, password, null, true);
 var hostUrl = Request.Url.GetComponents(UriComponents.SchemeAndServer,
UriFormat.Unescaped);
 var confirmationUrl = hostUrl +
VirtualPathUtility.ToAbsolute("~/Account/confirm?confirmationCode=" +
HttpUtility.HtmlEncode(token));

 WebMail.Send(to:email, subject:"Please confirm your account", body:"Your
confirmation code is " + token + ". Visit <a href=\"" + confirmationUrl + "\">" +
confirmationUrl + "</a> to activate your account.");

  Response.Redirect("~/Account/Thanks");

}

catch (MembershipCreateUserException ex){
 ModelState.AddFormError(ex.Message.ToString());
}
```

We specified null in the token this means that the extra information is not required. To add extra information about the user do the following:

1. In the database in the users table make more columns for data you want to collect.
2. Copy this code it goes above *var token*

```
var user = new{Name = name, Number = number, Age = age};
```

The Name is the column name and after the equal sign it's the HTML control getting this information. Now the token should look like this.

```
var token = WebSecurity.CreateUserAndAccount(email, password, user, true);
```

You will need to add additional HTML controls and also request the controls I am sure you know how to do this as we have been doing this for ages now!

REGISTRATION VIA USERNAME

Allowing users to register via a username is also an easy task, however, it also means you need to do extra work when you want to confirm the users account because you will need to access the database, get the email for the specified user and then send the email. If you decided to choose username as the registration method then you will need an additional column for email.

```
WebSecurity.InitializeDatabaseConnection("Membership","Users", "UserID", "Email",
true);
```

Now for this part all you need to do is change Email to **UserName** . Then run the website go in to the database and add another column named Email. You will also need to make sure that the Email column is unique, so you will need to compare the email the user enters through the database or make the column unique using a SQL constraint.

In the registration page here is what you need to add:

```
@{

if (IsPost){
var username = Request["username"];//add this

var user = new {Email = email};//add this
var token = WebSecurity.CreateUserAndAccount(username, password, user,
true);//change this
```

For the login page the process is the same – you need to change email and name it username.

```
var username = Request["email"];//change this

  if (WebSecurity.Login(username, password)){//change this
```

ACCOUNT LOCKED OUT STATUS

In the database in webpages_membership table there is a column named "PasswordFailureSinceLastSuccess" this counts the amount of password failures and once it reaches a number you specify the account is locked out. If you allow 5 attempts and the user logins in the 3rd attempt the value goes back to 0. In the account folder create a new file called **accountlockedout.cshtml** and insert some message saying the account is locked. Then in the login page above websecurity to login add this.

```
    if (WebSecurity.UserExists(email) &&
WebSecurity.GetPasswordFailuresSinceLastSuccess(email) > 5 &&
WebSecurity.GetLastPasswordFailureDate(email).AddMinutes(10) > DateTime.Now){
        Response.Redirect("~/account/accountlockedout.cshtml");
    }
```

We first simply check if the user exists, get the users password failures it takes one argument the username, then you are simply saying "password failure is above 5" then you get the password failure data and add minutes (or days if you want).

You then redirect the user, as you can see it is simple to lock accounts out on password failure.

2 STEP VERTIFICATION – USING TEXT LOCAL SMS GATEWAY

For those of you who are worried about security and being hacked, you can have a 2 step verification system where you login as usually but then get a 4 digit pin sent to your phone to enter on the site. We will use text local which is an SMS gateway. You get 10 free credits for first time users.

In the users table you need to add 3 more columns see below:

Column Name	Data Type
SecureCode	bigint
RequireSecureCode	Bit
Mobile	bigint

Now in the login page add this:

```
if (WebSecurity.Login(email, password)){

var db = Database.Open("Membership");
var SQLSELECT = "SELECT * FROM Users Where email=@0";
var data = db.QuerySingle(SQLSELECT, email);

var requireCode = data.RequireSecureCode;
var Mobile = data.Mobile;

      if (requireCode == true){

     Random myNumber = new Random();
     var SecureCode = myNumber.Next(1000, 9999);

var message= SecureCode.ToString();
var from = "";
var uname = "";
var hash = "";
var selectednums="";
var url = "";
var address = "https://www.txtlocal.com/sendsmspost.php";
var info = 1;
var test = 1;


message = HttpUtility.UrlEncode(message); //encode special characters (e.g. £, &
etc)
from = "";
uname = "";
hash = "";
selectednums = Mobile.ToString();
```

```
url = address + "?uname=" + uname + "&hash=" + hash + "&message=" + message +
"&from=" + from + "&selectednums=" + selectednums + "&info=" + info + "&test=" +
test;

HttpWebRequest code = (HttpWebRequest)HttpWebRequest.Create(url);
HttpWebResponse response = (HttpWebResponse)code.GetResponse ();
Stream receiveStream = response.GetResponseStream ();
StreamReader readStream = new StreamReader (receiveStream);

readStream.ReadToEnd();

var SQLUPDATE = "UPDATE Users set SecureCode=@0 WHERE email=@1";

try{
    db.Execute(SQLUPDATE, message, email);
}
catch (Exception ex){}

Response.Redirect("~/account/securecode.cshtml");

}
```

This is quite simple, we login as usual and once we have logged in we check to see if the user has two step verification enabled this is done by accessing the database and checking if the column requireSecureCode is true or false, if it is true then the code in the if statement executes, now this part of the code:

```
var message= SecureCode.ToString();
var from = "";
var uname = "";
var hash = "";
var selectednums="";
var url = "";
var address = "https://www.txtlocal.com/sendsmspost.php";
var info = 1;
var test = 1;


message = HttpUtility.UrlEncode(message); //encode special characters (e.g. £, &
etc)
from = "";
uname = "";
hash = "";
selectednums = Mobile.ToString();

url = address + "?uname=" + uname + "&hash=" + hash + "&message=" + message +
"&from=" + from + "&selectednums=" + selectednums + "&info=" + info + "&test=" +
test;
```

Is the text local API. It is a very simple API all it does it pass parameters in the URL so the URL will look like this https://www.txtlocal.com/sendsmspost.php?uname=yourusername?hash=yourhash

TextLocal API - Explained

Variables	Description
Message	This is the message you wish to send to the user, in this case it is a randomly generated number as string.
From	This is your company name or your name
Uname	This is your text local username
Hash	This is your text local has key, you can find this in your account – I highly recommend you use hash key instead of password.
Selectednums	This is the phone number you wish to send the text to, in this case it will be selected from the database, it needs the country code in front of it for example in UK it is 4471234567
url	The URL concatenates everything as you can see above
Address	The address is the txtlocal address
Info	By default it is set to 1 and I actually do not know what it does but according to the API documentation" it is to receive information about the request back to your application. "
Test	The test accepts either 0 or 1 as its value, 0 means that the application is not in test mode and your credit will go and 1 is the opposite ☺

Now copy the code, and enter your username and hash code, set test to 0.

The second part of the code executes the URL users cannot see this, it simply will execute the URL get the response and read the page data then message (random number) is saved it to the database and the user is redirected.

```
HttpWebRequest code = (HttpWebRequest)HttpWebRequest.Create(url);
HttpWebResponse response = (HttpWebResponse)code.GetResponse ();
Stream receiveStream = response.GetResponseStream ();
StreamReader readStream = new StreamReader (receiveStream);

readStream.ReadToEnd();

var SQLUPDATE = "UPDATE Users set SecureCode=@0 WHERE email=@1";

try{
    db.Execute(SQLUPDATE, message, email);
}
catch (Exception ex){}
```

```
Response.Redirect("~/account/securecode.cshtml");
```

Now in the account folder make another page called securecode.cshtml and copy this:

```
@{

 WebSecurity.RequireAuthenticatedUser();

    var db = Database.Open("Membership");
    var SQLSELECT = "SELECT SecureCode FROM Users Where UserId=@0";
    var data = db.QuerySingle(SQLSELECT, WebSecurity.CurrentUserId);

    var dbCode = data.SecureCode;

   long? code = null;

   if (IsPost){



if (Request["code"].IsEmpty()){
    ModelState.AddError("code", "Error: Enter Code");
} else{
    code = Convert.ToInt64(Request["code"]);
}

        if (code == dbCode){

            var sqlUpdate = "UPDATE Users Set securecode = null Where UserId=@0";

            try{
            db.Execute(sqlUpdate, WebSecurity.CurrentUserId);
            }
            catch (Exception ex){}

           Response.Redirect("~/members/default.cshtml");
       }else{
           <p>Error: Code is not valid!</p>
       }
   }

}


<form method="post">


<fieldset>
<legend>2 Step</legend>

<div><label>Enter Code</label></div>
    <input type="text" name="code" />
    @Html.ValidationMessage("code")

    <div>
```

```
    <input type="submit"/>
    </div>

</fieldset>
```

This is also very, very simple! At first you require an authenticated user, a user who has logged in by username and password and is on the second step, here we do the following:

1. Access the database select the SecureCode for the logged in user
2. Then we request the HTML control (code) – if it's empty show the first error otherwise convert it to a int64. (You won't be able to compare string and long so conversion is required)
3. Next compare the user entered code with the one stored in the database, if it matches the database updates itself and the column becomes null value.

Now in the members only folder (which you probably created if you were following the full tutorial) in the _PageStart.cshtml add this:

```
    var db = Database.Open("Membership");
    var SQLSELECT = "SELECT SecureCode FROM Users Where UserId=@0";
    var data = db.QuerySingle(SQLSELECT, WebSecurity.CurrentUserId);

    var code = data.SecureCode;

    if (code != null){
        Response.Redirect("~/account/securecode.cshtml");
    }
}
```

This simply checks the database and if the column securecode is not null (still has a value) for the logged in user then redirect the user to the securecode page otherwise allow them to enter.

That's it all done! You learned how to add additional user functionality, lock accounts and also a 2 step verification system. I hope you enjoyed this tutorial follow me on twitter, Facebook and check for regular updates at:

http://www.thecodingguys.net/downloads