

Exercise -Hashing Passwords using ASP.NET's Crypto Class (Assessable – Must Complete)

Background

Cryptography and protecting the data has always been the main point of interest for all computer programmers and enthusiasts. This enables them to secure their servers and to prevent any unauthorized data access. Usually, hackers attempt to gain access to a user's account data by logging into his account using his password. That is why it has always been a good approach to first encrypt the password and other sensitive information of a user's account and then store into the databases since SQL injection like methods can easily reveal the data stored in the database and hacker might be able to consume the information stored there.

Cryptography

Cryptography is a method used to protect the sensitive information and data from other parties that might use that data for any illegal activity.

Cryptography in ASP.NET

ASP.NET is a server-side programming language and provides a bunch of new namespaces for the programmers built on the .NET framework that makes it easy for the programmers to focus on the UI and UX of the web site and not the core features and processes that run the web site.

ASP.NET team has provided a new class as **Crypto** present inside the **System.Web.Helpers** namespace of [Web Pages](#) framework.

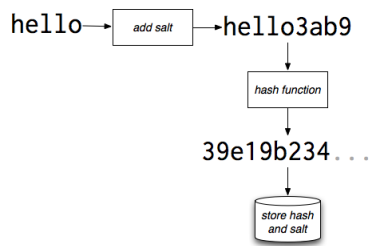
Hashing and Crypto Technique

It is worth noting and explaining what is hashing and how it is used to save the passwords.

Hashing is a process in which a password (from human understand form) is converted into a non-understandable form of **string**. That **string** is not directly, nor indirectly understandable by the humans. Hashing is used to change the password in any sense so that any one with rights to see the data in the database can never get the password to use the user's account for any purpose.



Salting is another technique used to make the hashing process faster. Salt is just a bunch of more characters that you add to the input before the hashing process takes place. This would create a much more strong hashing result and the **string** returned would be even stronger than before. But salting requires you to save the salt that was used while hashing the password since it cannot be regenerated.

It is also worth noting that once hashed, the **string** cannot be converted back to the original **string** that was passed at the time of hashing.



Salting is just an extra layer, that will be added to the password, as image shows that the salted password is not like the password that was sent. An extra character(s) is added to it. It plays its role for storing the same password's hash as a different hash value for different users. For example, in the following image, two same users use the same password "bob" but their salt; that was generated at their registration time, is different so the same password for them is saved differently.

Salting is used to minimize any errors or hacking issues that were caused by the attempt of an hacker to try out every possible permutation, combination of the characters in the English alphabets.

				
Password	bob	bob	bob	bob
Salt	-	-	et52ed	ye5sf8
Hash	f4c31aa	f4c31aa	lvn49sa	z32i6t0

Using the Methods

Crypto class exposes the following methods for working purposes in ASP.NET hashing process.

1. **string GenerateSalt()**

This method generates a new Salt to be added to the input **string** before the hashing process would start. This **string** needs to be saved because recreating of an exact match is almost impossible.

2. **string Hash()**

This function hashes the input **string** using either the default (SHA-256) algorithm or user can pass an algorithm for the ASP.NET to use to hash the password into.

3. **string HashPassword()**

This function returns an RFC 2898 hash value of the input **string** passed by the user.

4. **string SHA1()**

Returns the SHA1 hashed value for the input **string** provided.

5. **string SHA256()**

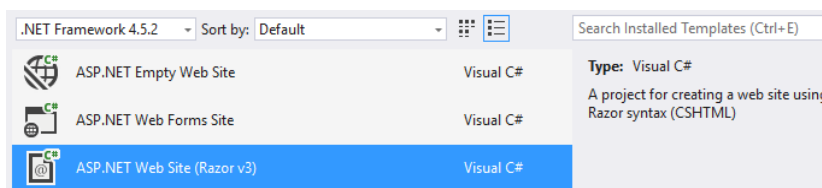
Same as the above, but the algorithm used is SHA-256.

6. `bool VerifyHashedPassword()`

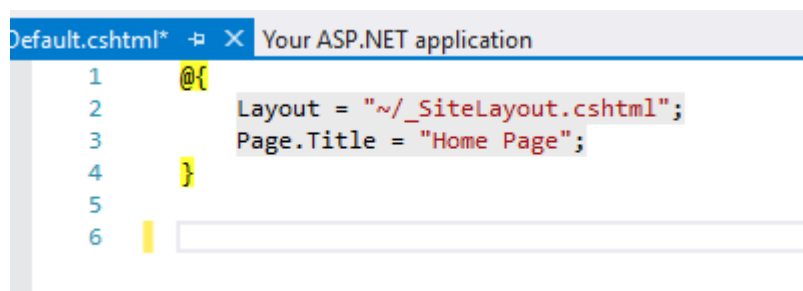
This method can be used by developers while authenticating the users. Because this method would check for the password sent by the user. Salt for the user would be saved in the database, and that salt would be added to the `Password string` provided by the user and then hashing would proceed resulting into the hashed value, if both values (the hashed value in database) and the value from user match then it returns `true`.

Lab Exercise

1. Create a new Web Site.. in VS 2015 (File -> New ->Project- > Web)
2. Select ASP.NET Web Site (Razor v3) -> OK (as shown below.)



3. Open the Default.cshtml page and delete all the code, except the top part of the page as shown below:



4. Replace the deleted code with the following code:

```

@{
    // Create the variables...
    // Remember: This password is just being shown to show the actual text being passed,
    // In real applications you shouldn't show the password to the User.
    var password = "";
    var hashed = "";
    var sha256 = "";
    var sha1 = "";

    var salt = "";

    var hashedPassword = "";
    var verify = false;

    // If the request is a POST request, then
    if (IsPost)
    {
        // Get the password
        password = Request.Form["password"];

        // Run the functions on the code,
        hashed = Crypto.Hash(password, "MD5");
        sha256 = Crypto.SHA256(password);
        sha1 = Crypto.SHA1(password);

        salt = Crypto.GenerateSalt();

        hashedPassword = Crypto.HashPassword(password);
    }
}

```

- Below the above code add the following html tags to create a UI to test your application:

```

<form method="post">
    <p>Write the string as a password that would be encrypted using
        <span style="color: #0094ff; font-family: Consolas;">
            Crypto</span> class of ASP.NET Web Pages.</p>

    <input type="password" name="password" autofocus />
    <input type="submit" value="Submit" />
</form>

<div>
    <p>Password: @password</p>
    <p>MD5 Hashed result: @hashed</p>
    <p>SHA256 result: @sha256</p>
    <p>SHA1 result: @sha1</p>
    <p>Salt: @salt</p>
    <p>HashedPassword: @hashedPassword</p>
    <p>Verify: @verify.ToString()</p>
</div>

```

6. Run the Default.cshtml page, enter a password and observe the different output of the hashed password using the different hashing algorithms (example shown below)

your logo here

Write the string as a password that would be encrypted using [Crypto](#) class of ASP.NET Web Pages.

Password: password123

MD5 Hashed result: 482C811DA5D5B4BC6D497FFA98491E38

SHA256 result: EF92B778BAFE771E89245B89ECBC08A44A4E166C06659911881F383D4473E94F

SHA1 result: CBFDAC6008F9CAB4083784CBD1874F76618D2A97

Salt: Kju1LtZ1NSyZIS2hjd004w==

7. Repeat the process using the same password. Add what you observed as a comment in your Default.cshtml page.
8. Zip up and upload your VS 2015 project using the Upload link on Learn