CSIS 4135

Authentication, Authorization, and Role Based Security

Security in .NET

- Can be integrated with Windows and NTFS security, or done independently
- Idea is to selectively restrict access to portions of a web site through
  - Authentication
  - Authorization
  - Impersonation
  - Delegation

ASP.NET Access Controls

- Authentication:
  - *Who are you?*
  - Verify that clients are who they say they are
- Authorization:
  - *What will I allow you to do?*
  - Does the client have permission to access the requested resource?
- Impersonation:
  - ASP.NET assumes the role of the user gaining access (has same access as user)
- Delegation:
  - More powerful form of impersonation

ASP.NET Security Layers

- Two Layers:
  - IIS can accept or reject a request
  - If accepted, request passed to ASP.NET which also makes a security decision
- IIS and ASP.NET security systems are completely independent of each other

Authentication

- Authentication required if access is to be restricted
- User credentials validated against some authority
  - Usually a user name and password
- Authority can be Windows security, or info stored in a config file or SQL Server
- If not done, client is an *anonymous* user
- Anonymous access is allowed by default

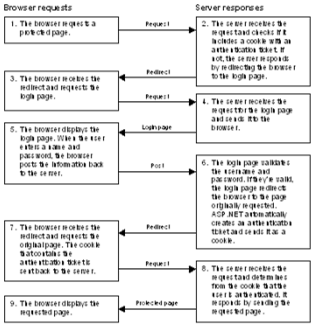
Authentication Modes

- Windows – Requires a Windows login account
- Form – HTML form gathers credentials and submits to the application
- Windows Live – Centralized Microsoft authentication service
- None

Forms Authentication

- Does not require clients to have a Windows account or recent version of Internet Explorer
- Programmer must do more work
  - Create custom login form
  - Compare user credentials with usernames and passwords in a data store
- Note: Form data sent unencrypted unless SSL is used

HTTP requests and forms-based authentication



Example

Login Form

Username:

Password:

Authenticate

(Bad) Example

```
private void Button1_Click(object sender, System.EventArgs e)
{
    if (userName.Text == "Mike" && password.Text == "password")
    {
        loginMsg.Text = "Authenticated!";
        FormsAuthentication.RedirectFromLoginPage(userName.Text, false);
    }
    else
    {
        loginMsg.Text = "Not authenticated";
    }
}
```

(Better) Example

```
private void Button1_Click(object sender, System.EventArgs e) {
    if (FormsAuthentication.Authenticate(userName.Text, password.Text))
    {
        FormsAuthentication.SetAuthCookie(userName.Text, false);
        Response.Redirect("welcome.aspx");
    }
    else
    {
        loginMsg.Text = "Not authenticated: " + userName.Text
                                              + " " + password.Text;
    }
}
```

Authentication data is in Web.config

Example

```
<authentication mode="Forms" >
    <forms name="QuoteASPCookie" loginUrl="login2.aspx" >
    <credentials passwordFormat="Clear">
    <user name="Tom" password="tommy" />
    <user name="Dick" password="dicky" />
    <user name="Harry" password="harry" />
    </credentials>
    </forms>
</authentication>
```

Membership API

- Implement Membership functionality using Login controls
- Role Based Security
  - Authentication: Using credentials to prove you are who you claim you are (user ID & password)
  - Authorization: User roles that give privileges for an authenticated user
    - Programmer can define the user roles

Site Admin

- Use Web Site Configuration tool, and select Security tab
- For Authentication type, select "From the Internet"
- Make a new role
- Create users (needs strong password)

Login page

- Add a page and put in a Login control
- Set the DestinationPageURL for where to go after logging in
- Can use LoginStatus, LoginName, LoginView controls

Login page

- Can also add a password recovery control, CreateUser or ChangePassword controls
- Use web site admin tool to see the users, delete, etc…

Roles Revisited

- Add a page for secure access only
- Also add a page for redirecting unauthorized access

Summary

- Membership class
  - Managing users
  - Working with lists of users
  - User stats
- Role class
  - Managing users in Roles
  - Managing Roles
- User class
  - Determining whether User is in a specific role