Module11

Controlling Access to ASP.NET MVC Web Applications

Module Overview

- Implementing Authentication and Authorization
- Assigning Roles and Membership

Lesson 1: Implementing Authentication and Authorization

- Local Authentication Providers
- Claims-Based Authentication
- Federated Authentication
- Restricting Access to Resources
- Demonstration: How to Authorize Access to Controller Actions

Local Authentication Providers

Local authentication providers include:

- ActiveDirectoryMembershipProvider:
  - Enables you to use only Active Directory as the user repository of your web application
- SqlMembershipProvider:
  - Works with a specific table schema that you can generate by using the **aspnet_regdb.exe** command
- SimpleMembershipProvider:
  - Requires only two key parameters, such as the user ID and the user name, and allows you to implement authentication that works with any database table schema
- UniversalProviders:
  - Works with any database supported by Entity Framework, but only with the database schema designed by Microsoft

Claims-Based Authentication

Claim-based authentication:

- Facilitates single sign-on
- Helps authenticate users
- Helps store user account information
- Integrates the application with the identity systems of other platforms or companies

Federated Authentication

Federated Authentication:

- Uses STS to:
  - Process claims from business partners
  - Extract information from claims
- Involves the FederatedPassiveSignIn Control to:
  - Exclude application-wide protection
  - Include a logon page with clickable controls
- Involves passive redirect to:
  - Verify the identity of the unauthenticated users
  - Issue security tokens that contain the appropriate claims for users

Restricting Access to Resources

The **Authorize** attribute:

- Restricts user access to information
- Mandates that users should be authorized to access information

The **AllowAnonymous** attribute:

- Allows users to access specific portions of information

Demonstration: How to Authorize Access to Controller Actions

In this demonstration, you will see how to:

- Generate authentication for a controller action
- Handle unauthenticated requests for actions that require authentication by using ASP.NET

Lesson 2: Assigning Roles and Membership

- Role Providers in ASP.NET 4.5
- Adding User Accounts to Roles
- Building a Custom Roles Provider
- Providing Membership Services
- Building a Custom Membership Provider
- Demonstration: How to Reset a Password

Role Providers in ASP.NET 4.5

Local role providers include:

- ActiveDirectoryRoleProvider:
  - Enables you to use only Active Directory as the management model for roles
- SqlRoleProvider:
  - Works with a specific table schema that you can generate by using the **aspnet_regdb.exe** command
- WindowsTokenRoleprovider:
  - Uses the Windows authentication token to determine the role of users
- SimpleRoleProvider:
  - Works with the various SQL databases to implement authorization based on tables defined by developers
- UniversalProviders:
  - Works with any database supported by Entity Framework, but only with the database schema designed by Microsoft

Adding User Accounts to Roles

To add users to roles:

- Implement the custom management tool in your application
- Use the **Authorize** attribute
- Ensure that the required role exists before using the **Authorize** attribute

Building a Custom Roles Provider

To build a custom role provider:

- Create a class that inherits the **RoleProvider** class
- Implement the **GetRolesForUser** function
- Modify the Web.config file

Providing Membership Services

To implement membership services in your web application:

- Modify the Web.config file
- Modify the AccountModel.cs file for any additional attributes
- Call the **WebSecurity.InitializeDatabaseConnection** function
- Add the **InitializeSimpleMembership** attribute in **AccountController** class

Building a Custom Membership Provider

To build custom membership providers:

- Implement the custom membership provider
- Override the **ValidateUser** function
- Override the **Provider** constructor to add additional logic

Demonstration: How to Reset a Password

- In this demonstration, you will see how to:
  - Create the code for the password reset operation.
  - Enable users to control their own password.

Lab: Controlling Access to ASP.NET MVC 4 Web Applications

- Exercise 1: Configuring Authentication and Membership Providers
- Exercise 2: Building the Logon and Register Views
- Exercise 3: Authorizing Access to Resources
- Exercise 4: Optional—Building a Password Reset View

Estimated Time: 90 minutes

Lab Scenario

A large part of the functionality for your proposed Photo Sharing application is in place. However, stakeholders are concerned about security because there are no restrictions on the tasks that users can complete. The following restrictions are required:

- Only site members should be able to add or delete photos.
- Only site members should be able to add or delete comments.

You have been asked to resolve these concerns by creating a membership system for the Photo Sharing application. Visitors should be able to register as users of the web application and create user accounts for themselves. After registration, when the users log on to the application, they will have access to actions such as adding and deleting photos and comments. Anonymous users will not have access to perform these actions. Additionally, registered users should also be able to reset their own password.

Lab Review

- In Exercise 3, when you tried to add a photo before logging on to the application, why did ASP.NET display the Login view?
- How can you ensure that only Adventure Works employees are granted access to the Delete action of the Photo controller?

Module Review and Takeaways

- Review Question(s)
- Real-world Issues and Scenarios