Microsoft

LESSON 5.1-5.2

10754 Microsoft .NET Fundamentals

Understand the System Security Namespace, Authentication, and Authorization

LESSON 5.1-5.2

10754 Microsoft .NET Fundamentals

Lesson Overview

Understand the System Security namespace, authentication, and authorization

In this lesson, you will:

- Learn about the role-based security model.
- Understand the difference between authentication and authorization.
- Learn about the code access security model.

Guiding Questions

1. What is role-based security?
2. How are roles used to determine if a person or user has access to a requested area or task?

Anticipatory Set

With a partner, brainstorm how you might manage the room security in a large office building shared by several companies.

- How would you determine which key each employee receives?
- How would you determine who would get a master key with access to every room in the building?

`System.Security` Namespace

- The `System.Security` namespaces contain classes that represent the Microsoft® .NET Framework security system and permissions.
- The classes in this namespace provide a framework for working with the common runtime language security system.
- One of the foundations of this security system is the concept of **permissions**.
- Note: although cryptography is not addressed specifically in this lesson, the `System.Security` namespace also provides cryptographic services.
  - Cryptography is the use of codes to convert ("encrypt") data so that only a specific recipient will be able to read it using a key. This is particularly useful when transmitting sensitive information over a network.

Permissions

- The .NET Framework allows code to perform only those operations that the code has permission to perform. It uses objects called *permissions* to enforce these restrictions.
  - .NET Framework provides built-in permission classes in several namespaces and also supports designing and implementing custom permission classes.
- There are two types of permissions, each with a specific purpose:
  - **Role-Based Security Permissions**—provides a mechanism for discovering whether a user has a particular identity or is a member of a specified role.
  - **Code Access Permissions**—represent access to a protected resource or the ability to perform a protected operation.

Role-Based Security

- Real-world security systems often use roles to determine the amount of access that various people have to a building.
- As an example, consider a bank.
  - A person classified as a "customer" has very limited access—he or she can probably only get into the lobby and other public areas, such as the restrooms.
  - A "teller" can access the public areas, plus the area behind the counter or window.
  - The "manager" can access all those areas, plus the safe.
- What roles are used at your school?
  - List the various roles that people are assigned and the building areas that each role can access.

Authentication and Authorization

- Role-based security depends on two processes: authentication and authorization.
- **Authentication** is the process of verifying the identity of the person or user, using the concept of *credentials*.
  - In the bank example, an employee's credentials might be a photo-ID card that is worn visibly at all times. Similarly, a school might use ID cards or uniforms to identify different roles.
  - In many applications, the credential is the user's password.
- **Authorization** is the process of determining whether a person or user is allowed to perform a task.
  - In the bank example, the employee ID might have a magnetic strip that is swiped to unlock a door. If the swiped card belongs to someone who is authorized to enter (based on that person's role), the door opens.

Role-Based Security in Applications

- Roles often are used in financial or business applications to enforce company policy.
  - Example: a financial application might impose limits on the size of the transaction being processed, depending on the role of the user making the request.
  - Clerks might have authorization to process transactions that are less than $10,000; supervisors might have a limit of $100,000; and presidents might have a still higher limit (or no limit at all).
- Role-based security also can be used when an application requires multiple approvals to complete an action.
  - Example: a purchasing system might allow any employee to generate a purchase request, but only a purchasing agent can convert that request into a purchase order or check.

Role-Based Security in the .NET Framework

- Role-based security in the .NET Framework supports authorization by making available information about the user (called the **principal**), which is constructed from an associated identity.
  - The identity (and the principal it helps to define) can be based either on a Microsoft Windows® account, or be a custom identity unrelated to a Windows account.
- Applications in the .NET Framework can make authorization decisions based on the principal's identity, role membership, or both.
  - A principal can be a member of one or more roles. Therefore, applications can use role membership to determine whether a principal is authorized to perform a requested action.
  - This means that even though a user is a member of a given role, he or she may be granted authorization to complete a specific task not allowed by his or her role. For example, a trustworthy bank teller might be given access to the safe.

`PrincipalPermission` Class

- The .NET Framework provides the `PrincipalPermission` class (in the `System.Security.Permissions` namespace) to handle role-based security.
- The `PrincipalPermission` class represents the identity or role that the principal must match.
- A principal's identity information can be accessed directly and role and identity checks performed in the code when needed.

Code Access Permissions

- In addition to role-based security, .NET uses **code access permissions**.
- Code access permissions are permission objects that help protect resources and operations from unauthorized use.
  - Rather than relying on roles to determine if a user has permission, this model is designed to prevent untrusted or unauthorized code from performing privileged or dangerous actions.
- Each code access permission represents one of the following rights:
  - The right to access a protected resource, such as files or environment variables
  - The right to perform a protected operation, such as accessing unmanaged code
- All code access permissions can be requested or demanded by code, and the Common Language Runtime decides which permissions, if any, to grant the code.

.NET Code Access Permissions (partial list)

- `FileDialogPermission`: Access files that have been selected by the user in an Open dialog box.
- `FileIOPermission`: Read, append, or write files or directories.
- `PrintingPermission`: Access printers.
- `RegistryPermission`: Read, write, create, or delete registry keys and values.
- `SecurityPermission`: Execute, assert permissions, call into unmanaged code, skip verification, and other rights.
- `SqlClientPermission`: Access Microsoft SQL Server® databases.
- `UIPermission`: Access user interface functionality.

For a complete list, see ***http://msdn.microsoft.com/en-us/library/h846e9b3.aspx***.

Ticket Out the Door

Answer the following questions:

1. Explain how a role-based security system works.
2. Explain the difference between authentication and authorization.
3. How is code access security different than role-based security?
4. What namespace represents the security and permission system used by the .NET Framework?