**보안 [ Security ]**

ASP.NET : 회원 가입, 로그인, 로그아웃등의 기능을 직접 구현

ASP.NET 2.0버전 : 보안 프로세스들이 모두 모듈화 되어 있다.

- 회원가입, 로그인, 로그아웃, 암호찾기와 같은 기능까지 로그인 컨트롤에서 지원된다.

**[ 인증과 권한 부여 ]**

- 인증 : ID, password를 통해 접근하는 로그인 과정

- 권한 부여 : 인증을 받은 사용자에 따라 특정 콘텐츠에 접근할 수 있는 유무를 줄 수 있다.

|  |  |
| --- | --- |
| 인증 방식 | 설명 |
| 윈도우 기반  (Windows-base) | Windows 의 권한 제어 방식을 웹 사이트에 그대로 적용  불특정 다수가 접속하는 인터넷 환경에서 모든 계정을 windows가 관리할 수 는 없슴  보통 윈도우즈 기반 인증은 주로 인트라넷 환경에서 사용됨 |
| 폼 기반  (Forms-based) | HTTP 쿠키와 HTML 폼을 이용하여 인증  폼 기반 인증은 ASP.NET에서 제공하는 인증 관련 API를 사용할 수 있기 때문에 인증 처리를 쉽게 할 수 있슴  참고로 ASP.NET 2.0에서 인증 관련 처리를 자동으로 해주는 맴버 자격은 폼 기반 인증을 사용함 |
| Passport | MS에서 제공하는 인증 서비스로 MS Passport의 파트너로 등록된 사이트에서 사용자 프로필을 공유하는 서비스  한 번의 로그인으로 여러 사이트에서 인증이 가능하도록 하는 서비스를 말함  하지만 상용 서비스라는 단점이 있슴 |
| 사용자 지정 | 인증 작업을 ASP.NET에서 제공하는 인증 방식을 사용하지 않고 인증과 관련된 프로세스를 개발자가 직접 구현함  개발자가 원하는 대로 세세한 구현을 할 수 있다는 장점이 있지만 다른 인증 방식에 비해 많은 개발 시간을 필요로 한다는 단점이 있슴 |

도구상자의 로그인 부분에 있는 컨트롤들을 사용하여 쉽게 구현할 수 있슴

**[ 폼기반 인증 ]**

폼 기반 인증은 쿠키와 ASP.NET 에서 제공하는 인증 관련 API를 사용해서 인증 처리를 하는 것을 말함

[ 도식화 ]

IIS

사용자 요청

로그인 페이지

인증

되었는가?

쿠키가 존재하는가?

쿠키 생성

권한이

부여되었는가?

요청 페이지

접근 거부

요청 페이지

접근 허가

제한된 페이지 요청 🡪 인증쿠키 존재 여부 확인을 통해 인증과정을 거침

쿠키가 존재하지 않는다면 페이지 접근 권한 여부를 묻게 되고 접근 권한이 있다면 최종적으로 요청한 페이지에 접근하게 됨

// Web.config

**<authentication mode="Forms">** : 폼 인증을 사용하기 위해 수정필요

: 기본모드는 Windows 모드임(즉 윈도우 인증 모드임)

<forms name=".ASPXAUTH" loginUrl="Ex01\_Login.aspx" defaultUrl="Ex01\_Default.aspx"></forms>

: 인증 권한을 가지고 있지 않은 사람일 경우 인증을 필요로 하는 페이지를 요청할 때

리디렉션할 페이지 URL을 지정함

default : 로그인 후 이동할 페이지 URL을 지정

<deny users="?"/> : 접근 권한에 대한 부분

users 특성에서 ?로 표현 : 익명 사용자를 의미하는 문자임

// Default.aspx

// 폼 기반 인증에서 인증된 사용자의 아이디는 Page.User.Identity.Name을 통해 가져올 수 있다.

protected void Page\_Load(object sender, EventArgs e)

{

Label1.Text = User.Identity.Name + " 님 환영합니다.!!!";

}

// 로그 아웃 버튼 클릭시 : 인증 쿠키 삭제 하고 로그아웃 처리 메서드 호출

protected void Button1\_Click(object sender, EventArgs e)

{

FormsAuthentication.SignOut();

Response.Redirect("Ex01\_Login.aspx");

}

// Login.aspx

인증 허용(인증 쿠키 생성 : 입력한 아이디와 암호가 모두 동일할 경우)

FormsAuthentication.RedirectFromLoginPage(TextBox1.Text, CheckBox1.Checked);

* (매개변수)사용자 이름, 쿠키 유지 여부를 나타내는 bool형식의 변수임
* 쿠키 유지 false이면 현 페이지에서만 인증이 허용

즉, 현 브라우저를 닫으면 인증 쿠키가 사라지고 로그아웃됨

쿠키 유지 여부 값을 true로 넘겨주면 웹 브라우저와 상관없이 한 번 인증된 내용이 계속적으로 유지됨

즉, 로그아웃 버튼을 통해 쿠키를 삭제하지 않는이상은 여러 브라우저를 띄우더라도 인증이 유지됨

**예제 확인 ) 폼 기반 인증**

**[권한 부여 ]**

폼 기반 인증에서 권한 부여는 Web.Config파일의 <authorization> 요소에서 지정함

* 접근 거부 : <deny>
* 접근 허가 : <allow>
* 사용자 : Users
* 역할 : Roles

Ex1) 모든 익명 사용자를 웹 사이트에 접근 못하게 하는<deny> 구문

< authorization>

<deny users="\*"/>

</authorization>

Ex2) 아이디가 “aaa”, “bbb” 인 사용자만 접근을 허용(allow)하고 나머지 모든 사용자들(\*)의 접근을 거부(deny)하

는 구문

< authorization>

<allow users="aaa,bbb"/>

<deny users="\*"/>

</authorization>

Ex3) ‘Admin’ 라는 역할을 가진 사용자들만 접근을 허용 <allow>하고 나머지 모든 사용자들(\*)의 접근을 거부하는(deny)하는 구문

< authorization>

<allow roles="Admins"/>

<deny users="\*"/>

</authorization>

Ex4) 특정 페이지에서만 권한 부여가 적용되도록 하려면?

<configuration>

<system.web>

<authentication mode="Forms">

<deny users="?"/> // 웹 사이트 전체에 적용하는 권한(모든 페이지에서

</authentication> // 익명 사용자 접근을 거부

</system.web>

**<location path="SuperAdminPage.asp">**

<system.web>

<authorization>

<allow roles="SuperAdmins"/> // SuperAdminPage 페이지 에서는 SuperAdmins역할을 가진

<deny users="\*"/> // 사용자만 접근 가능

</authorization>

</system.web>

</location>

**</configuration>**

**[ 로그인 컨트롤 사용하기 ]**

**1) 맴버 자격(Membership) 2.0 추가 컨트롤**

- 맴버 자격은 로그인 컨트롤, 맴버 자격 API, 맴버 자격 공급자가 서로 유기적으로 동작하도록

보안 프로세스(사용자 생성, 로그인, 로그아웃 등등)와 관련된 작업을 자동으로 처리해 주는 기능임

**로그인 컨트롤**

**맴버 자격 API**

**맴버 자격 공급자**

(SqlMembership Access 사용자지정

Provider) Membership Provider 맴버 자격 공급자

**데이터 소스**

SQL Server JET(Access) 사용자 지정 데이터 소스

\*) 맴버 자격 공급자 : 맴버 자격 API와 데이터 소스사이에 위치해 있슴

해당 위치에서 보안 프로세스와 관련된 작업들을 자동으로 처리함

\*) 개발자는 맴버 자격 API를 사용하여 프로그래밍 방식으로 맴버자격을 사용할 수 도 있고

맴버 자격 API를 사용하는 로그인 컨트롤을 통해 어떠한 코드도 작성하지 않고 맴버 자격을

사용할 수 있슴

**[ 로그인 컨트롤 ]**

|  |  |
| --- | --- |
| 로그인 컨트롤 | 설명 |
| CreateUserWizard(new 2.0) | 사용자 정보를 입력받아 새로운 사용자를 만들어 줌 |
| Login(new 2.0) | 로그인 기능을 제공함 |
| LoginView(new 2.0) | 인증된 사용자와 익명 사용자에게 보여줄 화면을 구분하여 나타냄 |
| LoginName(new 2.0) | 인증된 사용자의 이름을 표시 |
| LoginStatus(new 2.0) | 인증된 사용자에게는 로그아웃링크를 표시하고 익명 사용자에게는 로그인 링크를 표시함 |
| ChangePassword(new 2.0) | 암호 수정 기능을 제공 |
| PassWordRecovery(new 2.0) | 암호 찾기 기능을 제공 |

**CreateUserWizard 컨트롤**

* 사용자 정보를 입력받아 새로운 사용자를 만들어 주는 컨트롤
* 새로 생성된 사용자는 자동 처리됨

App\_Data 폴더에MDF 형태로 저장됨(SQL Server2005 Express 파일 기반 데이터베이스)

Machine.config,파일에서 맴버 자격 공급자가 사용하는 연결 문자열의 특성이 디폴트로 정해져 있기 때문에 생성됨

향후 Web.config파일에서 맴버 자격 공급자를 변경해 원하는 DB에 저장시킬 수 있슴

**Login 컨트롤**

* 로그인 기능을 제공하는 컨트롤
* 입력받은 사용자 이름 및 암호를 지정된 맴버 자격 공급자에게 넘겨줌

맴버 자격 공급자는 해당 데이터 소스에 접근하여 일치하는 사용자 정보를 검색함

일치하면 인증 쿠키를 생성하여 인증 처리를 하고, 존재하지 않는다면 해당 결과를 다시 Login컨트롤로 돌려주게 됨

\*) CreateUserWizard 컨트롤 및 Login 컨트롤의 사용 확인

**LoginView 컨트롤**

* 인증된 사용자와 익명 사용자에게 보여줄 화면을 구분해서 표시하는 컨트롤
* 생성된 기본 HTML코드는 아무런 기능이 없슴
* 필요한 기능은 컨트롤에서 제공하는 AnoymousTemplate과 LoggedInTempate을 추가하여 익명 사용자 및 인증된 사용자에게 보여줄 화면을 직접 디자인 해야 함

**LoginNam 컨트롤**

* 인증된 사용자의 이름을 출력하는 컨트롤
* 익명 사용자에게는 랜더링 되지 않음

**LoginStatus 컨트롤**

* 인증된 사용자에게는 로그아웃 링크를 익명 사용자에게는 로그인 링크를 표시하는 컨트롤

**ChangePassword 컨트롤**

* 암호 수정 기능을 제공하는 컨트롤
* 제대로 동작하기 위해서는 맴버 자격 공급자를 정의하는 <add>요소의 passwordFormat 특성이 Clear 또는 Encrypted인 상태에서 사용자를 생성해야 함
* <add>요소의 enablePasswordRetrieval 특성이 true로 지정되어 있어야 함
* 만얀 passwordFormat특성이 Hashed인 상태에서 사용자를 생성하면 암호가 해시 알고리즘으로 암호화 되어 다시 찾을 수 없게 됨

**PasswordRecovery 컨트롤**

* 잊어버린 암호를 사용자의 전자메일로 발송하는 컨트롤
* 제기능을 하려면 ChangePassword처럼 passwordFormat특성을 Clear 또는 Encrypted로 지정하고 enablepasswordRetrieval 특성을 true로 지정해야 함
* 찾은 암호를 사용자에게 전자메일로 발송하기 위해서는 웹 서버에 SMTP 서비스가 설치되어 있어야 함(메일 발송 프로토콜 규약)

**02) 로그인 컨트롤을 사용한 예제 확인**

**[ ASP.NET SQL Server 설치 마법사 사용하기 ]**

* 지금까지 사용했던 파일 기반 데이터베이스를 SQL Server2005 데이터 베이스에 구현
* Web.config 파일의 <ConnectionString>, <membership> 요소를 수정

<connectionString>

<add name=”MyLocalSqlServer” connectionString=”Data Source=.;Initial Catalog=aspnetdb;

User ID=sa;Password=sa”

providername=”System.Data.SqlClient” />

</connectionString>

<membership defaultProvider=”MyAspNetSqlMembershipProvider”>

<providers>

<add name=”MyAspNetSqlMembershipProvider”

connectionStringName=”MyLocalSqlServer”…//생략

</providers>

</membership>

**[ 사용자 지정 맴버 자격 공급자(Custom Membership Provider )**

ASP.NET 이 지원하는 기본적인 맴버 자격은 미리 정해진 정보를 입력받고 다룰 수 있게 함

추가적인 정보가 필요하다면?

결론은 맴버 자격 공급자를 사용자 지정하지 않는 이상 맴버 자격에서 새로운 정보를 추가할 수 없게됨

그러므로 로그인 컨트롤 및 맴버 자격 API에서도 새로운 정보를 처리할 수 있도록 하려면 SqlMemberShipProviser 클래스를 상속받아 다시 구현하거나, sqlmembershipProvider 클래스의 부모 클래스인 MembershipProvider 클래스를 상속받아 사용자 지정 맴버 자격 공급자를 구현해야 함

Public class MyMembershipProvider : System.Web.Security.MembershipProvier

{  
}

**[ 맴버 자격 API와 역할 관리자 ]**

로그인 컨트롤은 내부적으로 맴버 자격 API를 사용하여 관련 프로세스를 처리함

개발자는 맴버 자격 API를 소스코드에 직접 사용하여 맴버 자격을 프로그래밍 방식으로 제어할 수 있슴

두 개의 핵심 클래스

- Membership 클래스

- MembershipUser 클래스

이 두 클래스를 사용하면 맴버 자격에서 제공하는 사용자 생성, 사용자 정보 참조, 사용자 정부 수정,

사용자 유효성 검사, 사용자 삭제등등의 프로세스를 손쉽게 구현할 수 있다.

**예제 확인 : API를 이용한 사용자 생성**

MembershipCreateStatus status;

Membership.CreateUser(TextBox1.Text, TextBox2.Text, TextBox3.Text, TextBox4.Text, TextBox5.Text, true, out status);

switch (status)

{

case MembershipCreateStatus.Success :

Label1.Text = "회원 가입이 성공적으로 이루어졌습니다.!!!";

break;

case MembershipCreateStatus.InvalidUserName:

Label1.Text = "사용자 이름이 유효하지 않습니다";

break;

case MembershipCreateStatus.InvalidPassword:

Label1.Text = "암호가 유효하지 않습니다. 다시 입력하세요..";

break;

default :

Label1.Text = "회원 가입 도중 오류가 발생했습니다.";

break;

}

// **사용자 정보 참조** : 회원 정보 보기 페이지 및 회원 수정 페이지에서 회원 정보를 출력할 때 사용함

MembershipUser memUser = Membership.GetUser();

Response.Write(memUser.UserName + "<br>");

Response.Write(memUser.Email + "<br>");

Response.Write(memUser.LastActivityDate.ToString());

**// 사용자 정보 수정**

Membership클래스의 Update 메서드를 사용함

회원 수정페이지에서 회원 정보를 수정할 때 사용함

MembershipUser memUser = Membership.GetUser();

memUser.UserName = “수정된 사용자 이름”;

memUser.Email = “수정된 전자메일”;

…

Membership.UpdateUser(memuser);

**사용자 유효성 검사**

If( Membership.Validateuser(txtUserName.Text, txtPassword.Text))

{

FormsAuthentication.RedirectFromLoginPage(txtUserName.Text, false);   
}

else

{

Response.Write(“아이디가 존재하지 않거나 암호가 정확하지 않습니다.”);

}

**API를 이용한 사용자 삭제**

DeleteUser 메서드 사용

if (Membership.DeleteUser(user))

{

Label1.Text = user + " 계정이 정상적으로 삭제 되었습니다.";

BindDropDownList();

}

else

{

Label1.Text = user + " 계정 삭제 도중 오류가 발생했습니다.";

}