ASP.NET 2.0
Web Programming

이 현 정

hjyi@hotmail.com

MCT/MCSD/MCAD/MCSD.NET

목차

보안(13장)

- 인증과 권한 부여
  - ASP.NET의 인증 종류
- 로그인 컨트롤 사용하기
  - 멤버 자격(Membership)
  - 로그인 컨트롤들
  - ASP.NET SQL Server 설치 마법사 사용하기
- 멤버 자격과 역할 관리자
  - 멤버 자격 API
  - 역할 관리자

◆1.인증과 권한 부여

- 인증(**Authentication**)
  - Accepts credentials from a user
  - Validates the credentials
- 권한 부여(**Authorization**)
  - Given the authentication credentials supplied, determines the right to access a resource
  - Can be assigned by user name or by role

ASP.NET 인증 종류(p499)

Method	Advantages	Disadvantages
Windows-based Authentication	▪ Uses existing Windows infrastructure ▪ Controls access to sensitive information	▪ Not appropriate for most Internet applications
Forms-based Authentication	▪ Good for Internet applications ▪ Supports all client types	▪ Based on cookies
Microsoft Passport Authentication	▪ Single sign in for many Internet sites ▪ No need to maintain a database to store user information ▪ Allows developers to customize the appearance of the registration page	▪ Based on cookies ▪ Fees involved

1> Windows Authentication(IIS Authentication)

Mechanisms	Security Level	Description
Anonymous	None	▪ No authentication occurs
Basic	Low (Medium with SSL)	▪ Client sends username and password as clear text ▪ Can be encrypted by using SSL ▪ Part of the HTTP specification and supported by most browsers
Digest	Medium	▪ Sends information as encoded hash ▪ Requires Internet Explorer 5 or later ▪ Requires Active Directory
Integrated Windows	High	▪ Uses either NTLM or Kerberos ▪ Generally good for intranets, not Internet ▪ Does not work through most firewalls

Windows-Based Authentication 설정 방법

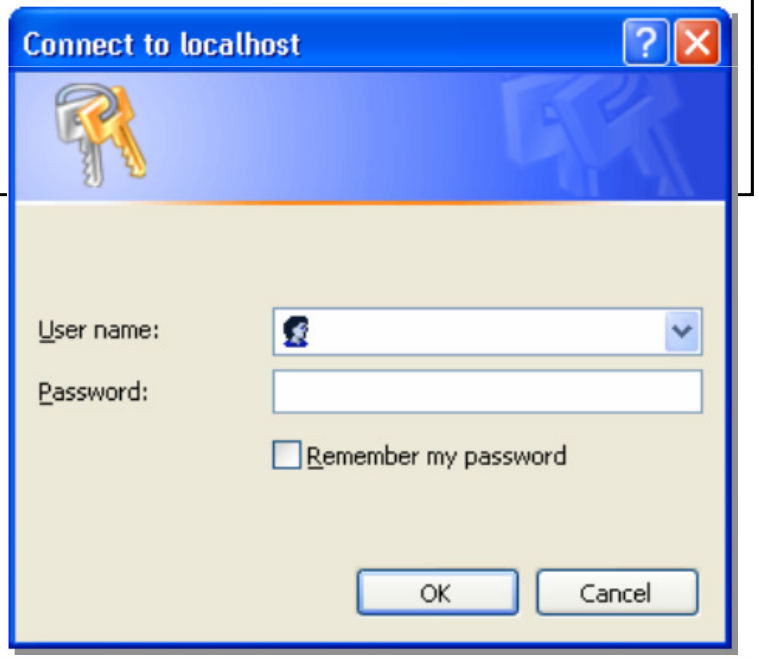
- **IIS를 구성 - 다음 중의 하나 또는 여러 개 선택**
  - 등록정보 – 디렉토리 보안 탭 – 익명 액세스및 인증제어에서 선택
  - 기본 인증(Basic)
  - 다이제스트 인증(Digest)
  - Windows통합 인증(Integrated Windows security)
- **Web.config에 인증 방법 명시(default)**

```
<system.web>
  <authentication mode="Windows" />
</system.web>
```

Windows-Based Authentication 설정 방법

- Web.config에 권한 부여(Authorization)을 설정

```
<location path="ShoppingCart.aspx">
   <system.web>
      <authorization>
         <deny users="?"/>
      </authorization>
   </system.web>
</location>
```



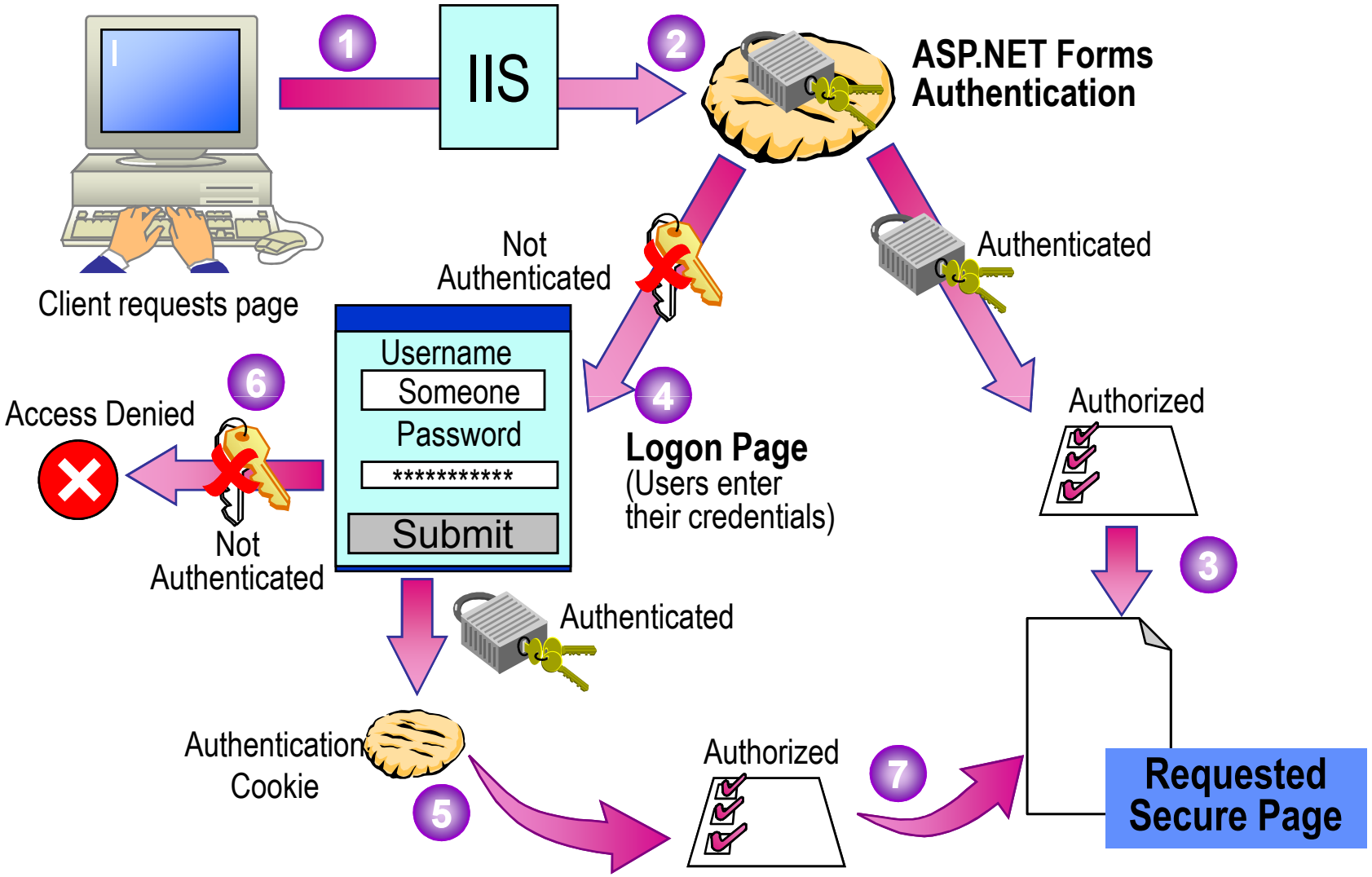
- When users access the Web Form, IIS requests logon information

Reading User Information

- After authentication, the Web server can read the user identity

```
lblAuthUser.Text = User.Identity.Name;
lblAuthType.Text = User.Identity.AuthenticationType;
lblIsAuth.Text = User.Identity.IsAuthenticated;
```

2> Forms-Based Authentication(p501)



Form-Based Authentication 설정

- **IIS – 익명 액세스만 선택**
  - 등록정보 – 디렉토리 보안 탭 – 익명 액세스및 인증제어에서 선택
- **Web.config를 수정**

```
<system.web>
  <authentication mode="Forms" />
   <forms name=".ASPXAUTH" loginUrl="loginDemo.aspx">
</system.web>
```

```
<authorization>
       <deny users="?"/>
</authorization>
```

3> 권한 부여(p506)

- **Web.config를 수정**

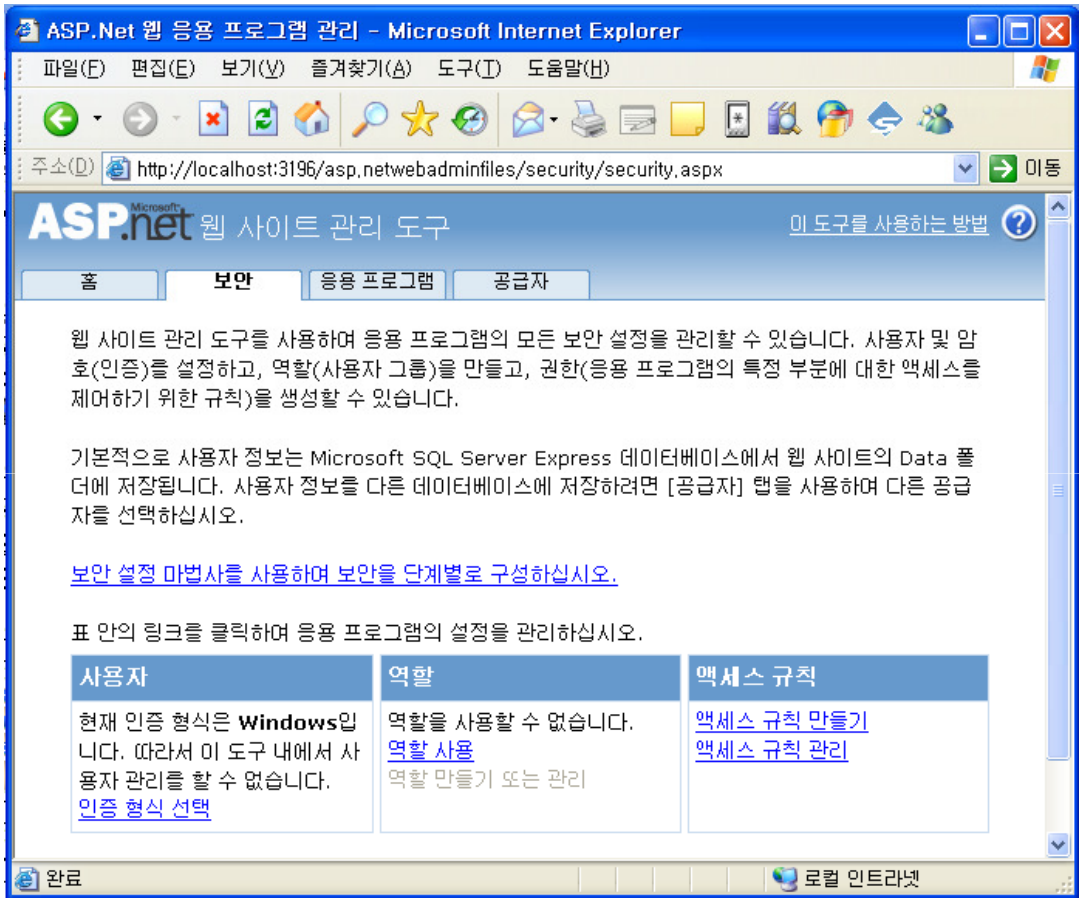
```
<authorization>
        <allow users="Mary,Kim"/>
        <deny users="?"/>
</authorization>
```

```
<authorization>
        <allow roles="Admins"/>
        <deny users="?"/>
</authorization>
```

```
<location path="SuperAdminPage.aspx">
   <system.web>
        <authorization>
           <allow roles="SuperAdmins"/>
           <deny users="?"/>
        </authorization>
   </system.web>
</location>
```

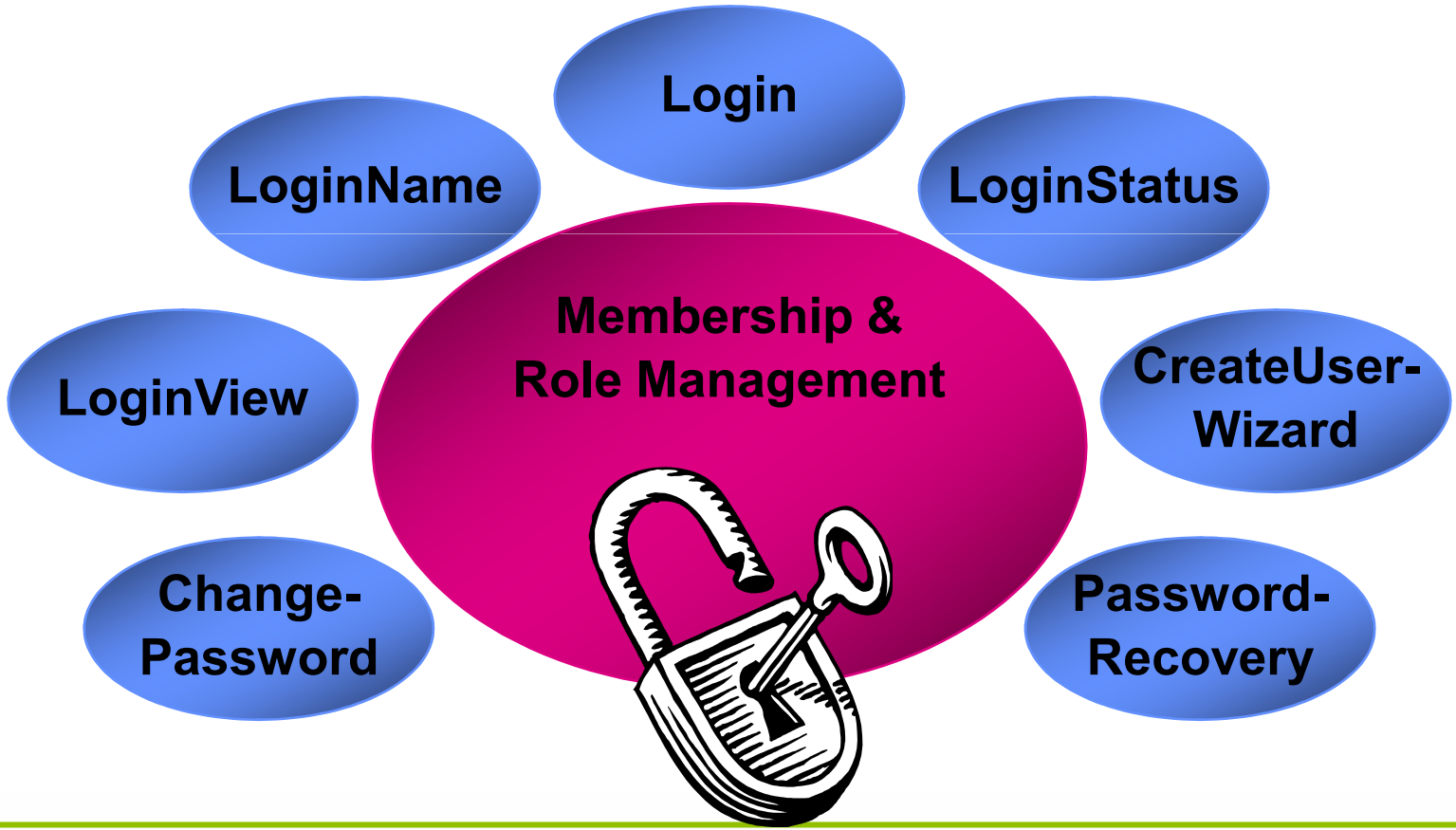
권한 부여

- 웹 사이트 관리 도구
  - Visual Studio2005
    -웹 사이트
    -ASP.NET 구성 메뉴

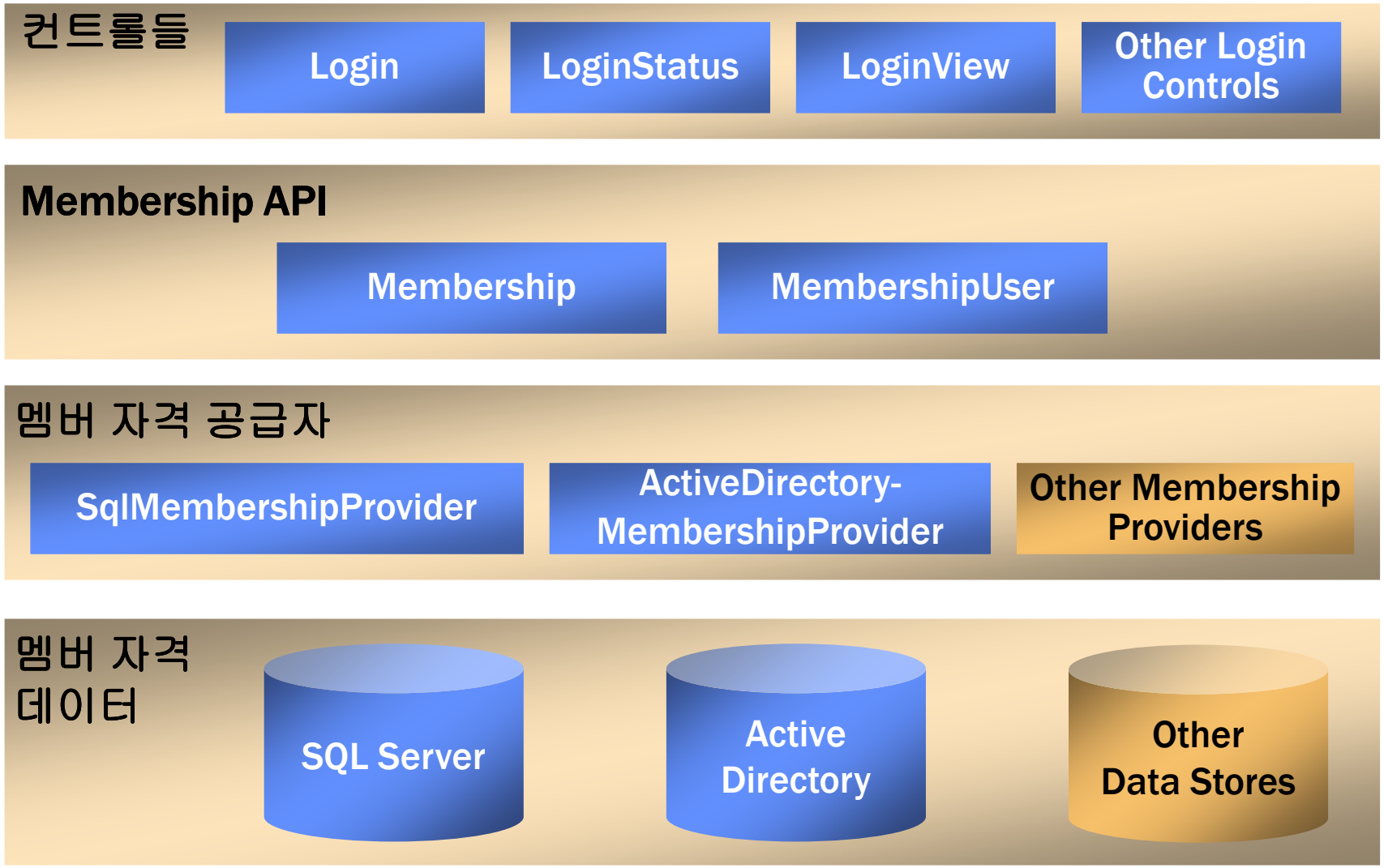


◆2. 로그인 컨트롤 사용하기(p508)

- 보안 관계 업무를 위한 UI와 로직



1> 멤버 자격(p509)


컨트롤들
Login
LoginStatus
LoginView
Other Login Controls
Membership API
Membership
MembershipUser
멤버 자격 공급자
SqlMembershipProvider
ActiveDirectory-MembershipProvider
Other Membership Providers
멤버 자격 데이터
SQL Server
Active Directory
Other Data Stores

멤버 자격 공급자 (기본값)- Machine.config(p511)


```
<configuration>
 <connectionStrings>
    <add name= "LocalSqlServer" connectionString="data source=.\SQLEXPRESS;
    Integrated Security=SSPI;AttachDBFilename=|DataDirectory|aspnetdb.mdf;
    User Instance=true" providerName="System.Data.SqlClient" />
  </connectionStrings>
```

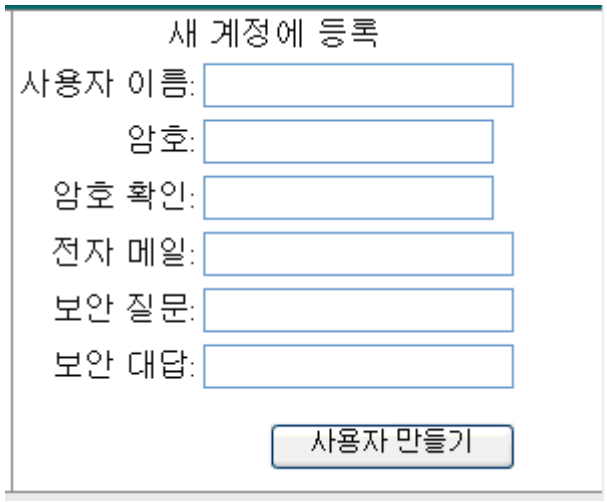
```
<membership>
  <providers>
     <add name=" AspNetSqlMembershipProvider "
      type="System.Web.Security.SqlMembershipProvider, System.Web, ..."
      connectionStringName="LocalSqlServer"
      enablePasswordRetrieval="false"
      enablePasswordReset="true"
      requiresQuestionAndAnswer="false"
      applicationName="/"
      requiresUniqueEmail="false"
      passwordFormat="Hashed"
      description="Stores and retrieves membership data ..."
    />
  </providers>
</membership>
```

Login 관련 컨트롤들(p513)

컨트롤	설명
ChangePassword	패스워드를 변경하는 UI
CreateUserWizard	새로운 사용자를 생성하는 UI
Login	사용자 이름과 패스워드를 입력하고 승인하는 UI
LoginName	인증받은 사용자 이름을 보여준다
LoginStatus	로그인 상태인지 로그아웃 상태인지를 보여줌
LoginView	로그인 상태와 Role에 근거해 다른 뷰를 보여줌
PasswordRecovery	패스워드 분실을 복구하는 UI

2> CreateUserWizard 컨트롤

- 사용자 정보를 입력받아 새로운 사용자를 만들어 주는 컨트롤
- Membership service와 통합됨
- Membership service없이도 사용 가능
- RequiredFieldValidators등 유효성 검사 컨트롤을 사용
- UI를 변경할 수 있다

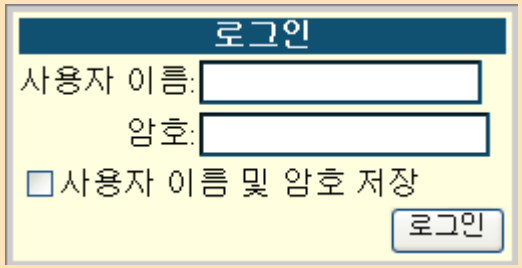


3> Login 컨트롤

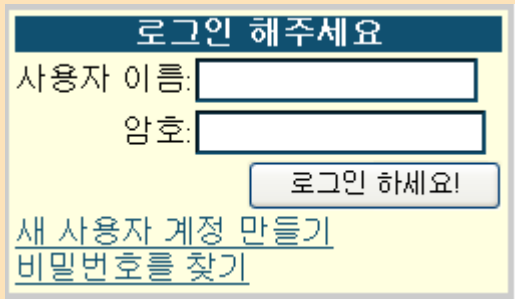
- 사용자 로그인을 위한 표준 UI 제공
- Membership service와 통합됨
  - 자동적으로 ValidateUser()를 호출
  - 로그인과 승인 관련해 코드를 작성할 필요가 없다
- Membership service없이도 사용 가능
- RequiredFieldValidators 사용가능
- UI를 변경할 수 있다

Login 컨트롤 사용하기


```
<html>
  <body>
    <form runat="server">
      <asp:Login RunAt="server" />
    </form>
  </body>
</html>
```




```
<asp:Login ID="LoginControl" runat="server"
  CreateUserText="새 사용자 계정 만들기"
  CreateUserUrl="CreateUser.aspx"
  DisplayRememberMe="False"
  LoginButtonText="로그인 하세요!"
  PasswordRecoveryText="비밀번호를 찾기"
  PasswordRecoveryUrl="RecoverPassword.aspx"
  TitleText="로그인 해주세요">
</asp:Login>
```



Login 컨트롤 이벤트

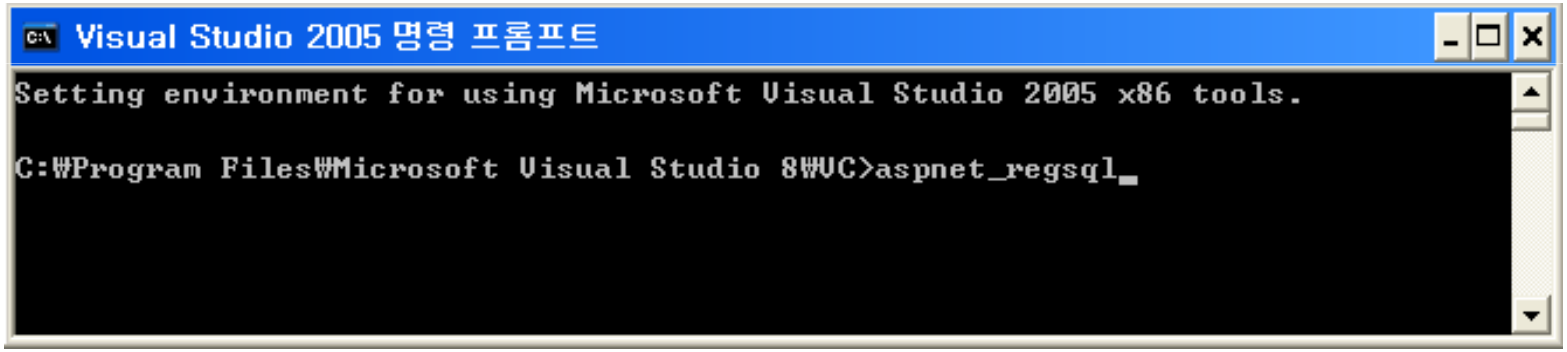
이름	설명
Authenticate	Log In Button을 클릭할 때 발생한다. 목적:사용자가 제시한 로그인 정보(credentials)로 유효한 사용자인지를 인증
LoggedIn	성공적으로 로그인이 되면 발생
LoggingIn	사용자가 Log In button를 클릭할 때 발생한다. 목적: 로그인 정보(credentials)에 대한 유효성 검사를 미리함 (예. 이메일주소가 이메일주소형식에 맞는 지를 검사)
LoginError	로그인에 실패했을 때 발생

4> 기타 Login 관련 컨트롤들

컨트롤	설명
LoginView	로그인 상태와 Role에 근거해 다른 뷰를 보여줌
LoginName	인증받은 사용자 이름을 보여준다
LoginStatus	로그인 상태인지 로그아웃 상태인지를 보여줌
ChangePassword	패스워드를 변경하는 UI
PasswordRecovery	패스워드 분실을 복구하는 UI

5> ASP.NET SQL Server 설치 마법사 사용하기

- **ASPNETDB를 먼저 생성해야 됨**
  - 시작메뉴-> 모든 프로그램->Microsoft Visual Studio 2005->Visual Studio Tools->Visual Studio 명령 프롬프트에서 aspnet_regsql를 입력



멤버 자격 공급자(Membership Providers)

- **멤버 자격 관련 정보는 공급자(Provider)기반으로 저장됨**
  - 공급자는 멤버 자격 서비스와 실제 데이터 저장소간의 인터페이스 역할을 함
- **Visual Studio 2005**
  - SqlMembershipProvider (SQL Server)
- **Custom providers를 사용하여 다른 데이터 저장소에 저장**

공급자(Provider) 구성

- 멤버 자격 공급자는 여러 가지 구성 정보 셋팅을 제공한다
  - 패스워드가 어떻게 저장될 것인가?
    cleartext, hashed, encrypted
  - 패스워드 복구가 가능하게 할 것인가?
  - 각 사용자가 유일한 이메일 주소를 가지게 할 것인가?
- Provider 클래스의 속성으로 제공
- .CONFIG 파일

공급자 셋팅 변경하기


```
<configuration>
 <connectionStrings>
    <add name= "MyLocalSqlServer" connectionString="data source=localhost;
       Initial Catalog=aspnetdb;Integrated security=true"
       providerName="System.Data.SqlClient" />
  </connectionStrings>
```

```
<membership>
  <providers defaultProvider="MyAspNetSqlMembershipProvider">
    <add name="MyAspNetSqlMembershipProvider "
      type="System.Web.Security.SqlMembershipProvider, System.Web, ..."
      connectionStringName="MyLocalSqlServer"
      enablePasswordRetrieval="false"
      enablePasswordReset="true"
      requiresQuestionAndAnswer="false"
      applicationName="/"
      requiresUniqueEmail="false"
      passwordFormat="Hashed"
      description="Stores and retrieves membership data ..."
    />
  </providers>
</membership>
```

◆3. 멤버 자격 API와 역할 관리자

- 멤버 자격 API
  - Membership 클래스
  - MembershipUser 클래스
- 역할 관리자

1> 멤버 자격 API

- 사용자와 인증정보(Credentials)를 관리해주는 서비스
  - 웹 사이트 관리 도구를 사용하여 선언적으로 액세스
  - Membership과 MembershipUser 클래스를 사용하여 프로그래밍으로 액세스
- **Membership 클래스로 기본 서비스를 제공**
- **MembershipUser 클래스는 사용자와 추가적인 서비스를 제공**
- 공급지(Provider)기반의 데이터 저장소 사용

Membership 클래스

- 멤버 자격 관련 일을 수행하는 정적 메소드를 제공
  - 사용자 생성 및 삭제
  - 사용자에 대한 정보 추출
  - 랜덤 패스워드 생성
  - 로그인 승인
- 공급자(Provider) 셋팅에 관련된 정보를 얻는 읽기 전용 정적 속성 제공

Membership 속성들

이름	설명
EnablePasswordReset	현재 멤버 자격 공급자에서 사용자가 암호를 다시 설정하도록 구성되어 있는지 여부를 나타내는 값을 가져옵니다.
EnablePasswordRetrieval	현재 멤버 자격 공급자에서 사용자가 암호를 찾도록 구성되어 있는지 여부를 나타내는 값을 가져옵니다.
MaxInvalidPasswordAttempts	멤버 자격 사용자가 잠겨지기 전에 허용되는 잘못된 암호 또는 암호 대답 시도 횟수를 가져옵니다.
MinRequiredPasswordLength	암호에 필요한 최소 길이를 가져옵니다.
Provider	응용 프로그램의 기본 멤버 자격 공급자에 대한 참조를 가져옵니다.
RequiresQuestionAndAnswer	기본 멤버 자격 공급자에서 사용자가 암호 재설정 및 찾기를 위해 암호 질문에 대답해야 하는지 여부를 나타내는 값을 가져옵니다.
UserIsOnlineTimeWindow	온라인인 것으로 간주되는 사용자에 대해 마지막 작업 날짜/시간 스탬프 이후의 시간(분)을 지정합니다.

Membership 메소드

이름	설명
CreateUser	Membership 데이터 저장 장소에 사용자를 추가
DeleteUser	Membership 데이터 저장 장소에 사용자를 제거
GeneratePassword	지정된 길이의 랜덤 패스워드를 생성
GetAllUsers	현재 등록된 모든 사용자들을 나타내는 MembershipUser개체들에 대한 컬렉션을 가져온다
GetUser	사용자에 대한 MembershipUser 개체를 가져온다
UpdateUser	지정된 사용자에 관련된 정보를 업데이트
ValidateUser	사용자 이름과 패스워드에 근거한 로그인에 대한 유효성을 검사

Membership 메소드

이름	설명
FindUsersByEmail()	전자 메일 주소가 지정한 전자 메일 주소와 일치하는 멤버 자격 사용자의 컬렉션을 가져옵니다.
FindUsersByName()	사용자 이름이 지정한 사용자 이름과 일치하는 멤버 자격 사용자의 컬렉션을 가져옵니다.
GetNumberOfUsersOnline()	현재 응용 프로그램에 액세스하는 사용자 수를 가져옵니다.

사용자 생성(p531)

C#

```
try
{
    Membership.CreateUser (“admin", “P@sswOrd", “admin@microsoft.com");
}
catch (MembershipCreateUserException e)
{
    // Find out why CreateUser failed
    switch (e.StatusCode) {

    case MembershipCreateStatus.DuplicateUsername:
      ...
    case MembershipCreateStatus.DuplicateEmail:
      ...
    case MembershipCreateStatus.InvalidPassword:
      ...
    default:
      ...
    }
}
```

사용자 유효성 검사(p533)

- 로그인 승인

```
if (Membership.ValidateUser (txtUserName.Text, txtPassword.Text))
    FormsAuthentication.RedirectFromLoginPage (txtUserName.Text,
        RememberMe.Checked);
```

- 사용자 삭제

```
if (Membership.DeleteUser(User.Identity.Name))
{
  FormsAuthetication.SignOut();
  Response.Redirect("~/Default.aspx");
}
else
  lblResult.Text = "회원 삭제가 정상적으로 이루어지지 않았습니다.";
```

MembershipUser 클래스

- 멤버 자격 데이터 저장소에 등록되어있는 각각의 사용자를 나타냄
- 사용자 정보를 얻거나 변경하기 위한 속성들을 제공
- 패스워드를 가져오거나 변경, 리셋하기 위한 메소드들을 제공
- **GetUser()**나 **CreateUser()**에 의해 리턴됨

MembershipUser 속성들

이름	설명
Comment	사용자 정의 데이터에 대한 저장
CreationDate	Membership 데이터 저장 장소에 사용자가 추가된 날짜
Email	사용자의 이메일 주소
LastLoginDate	사용자가 성공적으로 마지막 로그인한 날짜
LastPasswordChangedDate	사용자의 패스워드가 마지막으로 변경된 날짜
UserId	Membership Provider에 의해 생성된 유일한 UserId
UserName	등록된 사용자 이름

MembershipUser 메소드

이름	설명
ChangePassword	사용자의 패스워드를 변경
ChangePassword-QuestionAndAnswer	패스워드 복구시 사용되는 질문과 답을 변경
GetPassword*	패스워드를 가져온다
ResetPassword	새로운 랜덤 패스워드로 패스워드를 다시 설정

** Membership.EnablePasswordRetrieval 이 true 인 경우에만 동작한다*

사용자 정보(p532)

■사용자 정보 참조

```
MembershipUser memuser = Memebrship.GetUser();
Response.Write(memUser.UserName + “<br>”);
Response.Write(memUser.Email + “<br>”);
Response.Write(memUser.LastActivityDate.ToString());
```

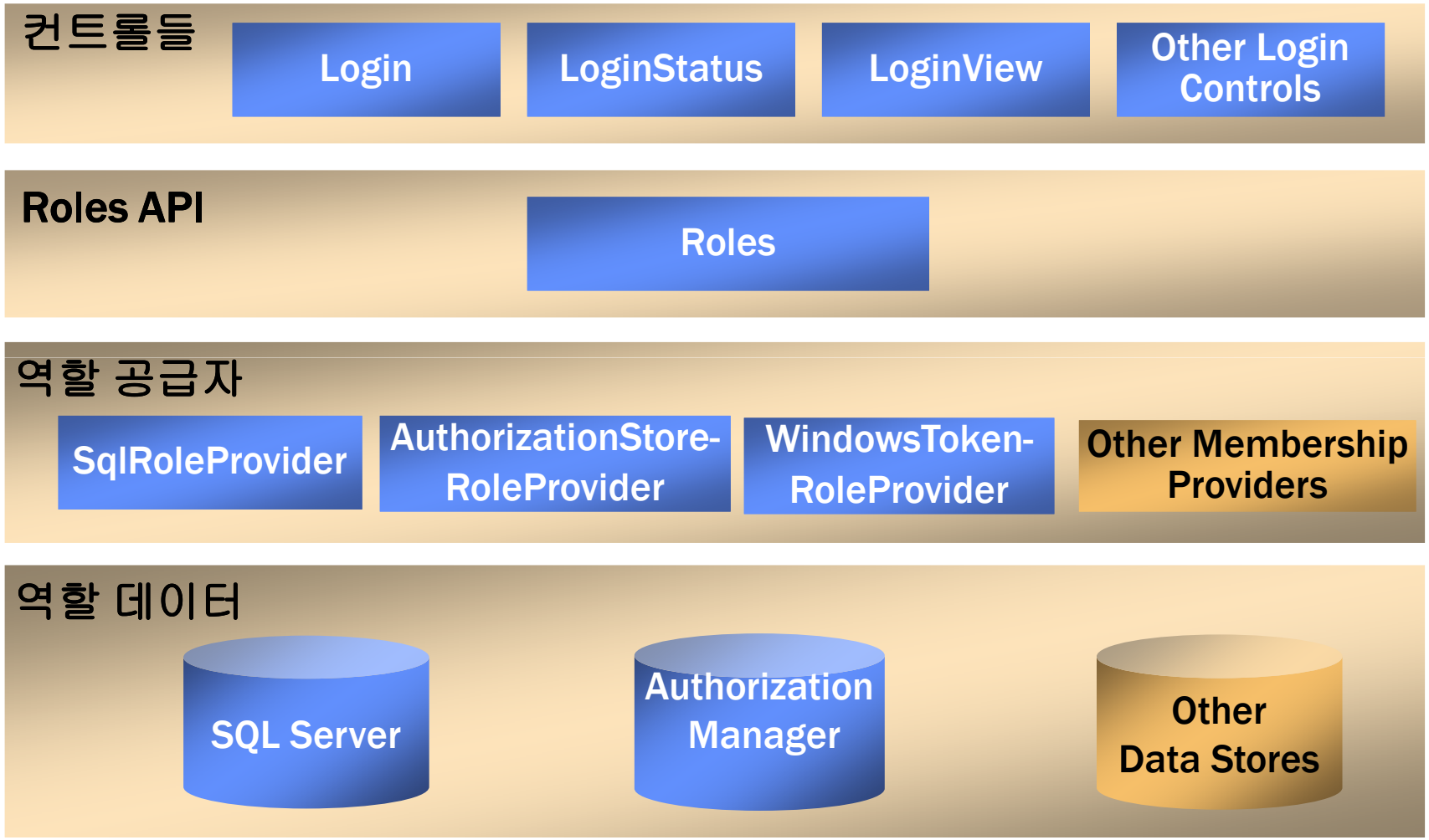
■사용자 정보 수정

```
MembershipUser memuser = Memebrship.GetUser();
memUser.UserName = “수정된 사용자 이름”;
memUser.Email =“수전된 전자 메일”;
//….
Membership.UpdateUser(memUser);
```

2> 역할 관리자(Role Manager)(p534)

- 역할 기반 보안(Role-based security)
  - 웹 사이트 관리 도구를 사용하여 선언적으로 액세스
  - Roles 클래스를 사용하여 프로그래밍으로 액세스
- Roles클래스는 새로운 역할을 생성하고 역할에 사용자를 추가하는 메소드를 제공
- 각 Request마다 사용자를 해당 역할에 맵핑시킴
  - ASP.NET1.x의 Application_AuthenticateRequest()
- 공급자(Provider) 기반의 데이터 저장소 사용
  - 기본역할공급자 : AspNetSqlRoleProvider(SQL 2005 Express)

역할 관리 스키마


컨트롤들
Login
LoginStatus
LoginView
Other Login Controls
Roles API
Roles
역할 공급자
SqlRoleProvider
AuthorizationStore-RoleProvider
WindowsToken-RoleProvider
Other Membership Providers
역할 데이터
SQL Server
Authorization Manager
Other Data Stores

역할 관리자 활성화(p535)

- 역할 관리는 기본적으로는 비활성화되어있다
- 웹 사이트 관리도구나 **Web.config**로 활성화시킬 수 있다

```
<configuration>
  <system.web>
    <roleManager enabled="true“ defaultProvider=“MyAspNetSqlRoleProvider” />
    <providers>
        <add name=“MyAspNetSqlRoleProvider” type=“System.Web.SqlRoleProvider”
          ….
         connectionStringName=“LocalSqlServer”/>
    <providers>
  </system.web>
</configuration>
```

Roles 클래스

- 역할 관리 API
- 역할 관리에 관련된 일을 수행하는 정적 메소드를 제공
  - 역할 생성 및 삭제
  - 역할에 사용자 추가
  - 역할에서 사용자 삭제
- 공급자 셋팅에 관련된 정보를 얻는 읽기 전용 정적 속성 제공

Roles 속성

이름	설명
CacheRolesInCookie	현재 사용자의 역할이 쿠키에 캐시되는지 여부를 나타내는 값을 가져옵니다
CookieName	역할 이름이 캐시되는 쿠키의 이름을 가져옵니다.
CookiePath	캐시된 역할 이름 쿠키에 대한 경로를 가져옵니다.
CreatePersistentCookie	역할 이름 쿠키가 세션 기반인지, 아니면 영구적인지 여부를 나타내는 값을 가져옵니다.
CookieRequireSSL	서버에 반환되기 위해 역할 이름 쿠키에 ***SSL***이 필요한지 여부를 나타내는 값을 가져옵니다
CookieSlidingExpiration	역할 이름 쿠키 만료 날짜 및 시간을 정기적으로 다시 설정하는지 여부를 나타냅니다.
CookieTimeOut	역할 쿠키가 만료되기까지의 시간*(분)*을 가져옵니다.

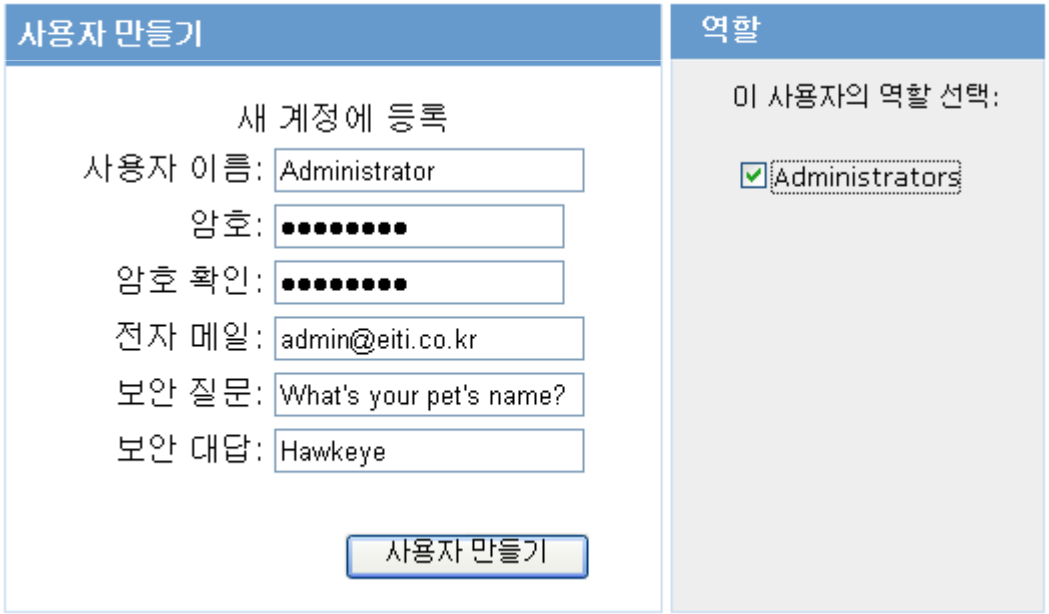
Roles 메소드(p536)

이름	설명
AddUserToRole	Role에 사용자를 추가
CreateRole	새로운 Role를 생성
DeleteRole	기존의 Role를 삭제
GetRolesForUser	사용자가 속해있는 Role들에 대한 컬렉션을 가져온다
GetUsersInRole	지정된 Role에 속해있는 사용자들에 대한 컬렉션을 가져온다
IsUserInRole	사용자가 명시된 Role에 속해있는 지 여부를 나타낸다
RemoveUserFromRole	명시된 Role에서 사용자를 제거한다

새로운 역할 생성하고 사용자 추가하기


```
if (!Roles.RoleExists ("Developers")) {
    Roles.CreateRole ("Developers");
}

string name = Membership.GetUser().UserName;
Roles.AddUserToRole(name,"Developers");
```



14장 프로필

- 프로필(Profile) 들어가기
  - 프로필이란?
  - 프로필 구조
- 프로필 사용하기
  - 프로필 속성 선언하기
  - 익명사용자를 위한 프로필
  - 프로필 이동

◆1. 프로필 들어가기(p541)

- 프로필*(Profile)* 이란*?*
  - **사용자와 관련된 정보를 사용자별로 지속적으로 저장**
    - 강력한 타입으로 액세스
    - 요구시에만 조회(On-demand lookup)
    - 오랫동안 남아있다(Long-lived)
    - 인증된 사용자뿐 만 아니라 익명 사용자도 지원
  - **동적으로 생성된 ProfileBase 파생 클래스(ProfileCommon)로 액세스**
    - HttpContext.Profile속성으로 액세스
  - **공급자(Provider) 기반의 데이터 저장소 사용**
    - SqlProfileProvider - AspNetSqlProfileProvider

◆2.프로필 사용하기(p544)

■ 프로필 속성 선언 - Web.config에 명시

- Type
  - 디폴트는 String
  - 문자열, 숫자 값 또는 **DateTime** 값 등의 기본 데이터 형식
  - 사용자 정의 형식

```
<configuration>
  <system.web>
    <profile>
      <properties>
        <add name="MainBackColor" />
        <add name="EnableCalendar" type="System.Boolean" />
        <group name="MemInfos">
            <add name="Nickname">
            <add name="Age" type="System.Int32"/>
        </group>
      </properties>
    </profile>
  </system.web>
</configuration>
```

프로필 사용하기

- **Profile속성**
  - ASP.NET에 의해 생성되는 클래스에서만 사용(.aspx,.asax등)
  - Profile.PropertyName : 현재 로그온 한 사용자
  - Profile.GetProfile("username") : 다른 사용자
- **HttpContext.Profile 속성**
  - 외부 컴포너트에서 사용

```
string color = Profile.MainColor;

// Get a reference to Fred's profile
ProfileCommon profile = Profile.GetProfile("Fred");

// Read the current user's ScreenName property in an external component
string color=(string)HttpContext.Current.Profile["MainColor"];

// Group으로 저장된 Profile 읽어오기
string name2 = Profile.MemInfos.NickName;
```

프로필 공급자

- 프로필 관련 정보는 공급자 기반으로 저장됨
- **Visual Studio 2005**
  - SqlProfileProvider (SQL Server)
    : aspnet_regsql.exe로 ASPNETDB 생성 한 후 사용
- **Custom providers를 사용하여 다른 데이터 저장소에 저장**

```
<configuration>
  <system.web>
    <profile defaultProvider="MyAspNetSqlProfileProvider" >
       <providers>
         <add name=" MyAspNetSqlProfileProvider "
          connectionStringName="MyLocalSqlServer" applicationName="/"
          type="System.Web.Profile.SqlProfileProvider,
            System.Web, Version=2.0.0.0, Culture=neutral,
            PublicKeyToken=b03f5f7f11d50a3a" />
        </providers>
  </system.web>
</configuration>
```

◆익명 사용자를 위한 프로필(p552)

- 디폴트는 익명 사용자에 대해서는 프로필을 사용할 수 없다
  - 인증받은 사용자 ID로 데이터가 입력됨
- 익명사용자 프로필을 활성화할 수 있다(web.config)
  - Step 1: Anonymous identification을 활성화
  - Step 2: Profile Property에 익명 사용자가 사용할 수 있도록 명시
- 익명 사용자ID로 데이터가 입력됨

```
<configuration>
  <system.web>
    <anonymousIdentification enabled="true" />
    <profile>
      <properties>
        <add name="MainBackColor" allowAnonymous="true" />
        <add name="EnableCalendar" type="System.Boolean"
             allowAnonymous="true" />
      </properties>
    </profile>
  </system.web>
</configuration>
```

프로필 이동(익명 사용자-> 인증된 사용자)

- 사용자가 로그인하는 경우, MigrateAnonymous 이벤트가 발생함
  - 사용자의 익명 ID에서 가져온 정보를 인증된 새 ID로 마이그레이션할 수 있습니다.

Global.asax

```
<script runat="server">
public void Profile_MigrateAnonymous (Object sender,ProfileMigrateEventArgs e)
{
    //익명 사용자를 위한 셋팅을 바꾸기위해 AnonymousIdentification_Remove 다음에 호출됨
    ProfileCommon anonProfile = Profile.GetProfile(e.AnonymousID);
    Profile.MainBackColor = anonProfile.MainBackColor;
    Profile.EnableCalendar = anonProfile.EnableCalendar ;

    //익명 사용자 프로필 및 익명사용자 ID를 삭제한다.
    ProfileManager.DeleteProfile(e.AnonymousID);
    AnonymousIdentificationModule.ClearAnonymousIdentifier();
}
</script>
```

프로필 API

- **ProfileManager클래스**
  - 프로필 정보를 관리
  - 특정 사용자의 프로필을 삭제하거나 참조할 수 있다.
- **ProfileManger Methods**
  - DeleteProfile()
  - DeleteProfiles()
  - DeleteInactiveProfiles()
  - FindProfilesByUserName()
  - FindInactiveProfilesByUserName()
  - GetAllProfiles()
  - GetAllInactiveProfiles()

실습 14 : 프로필 I

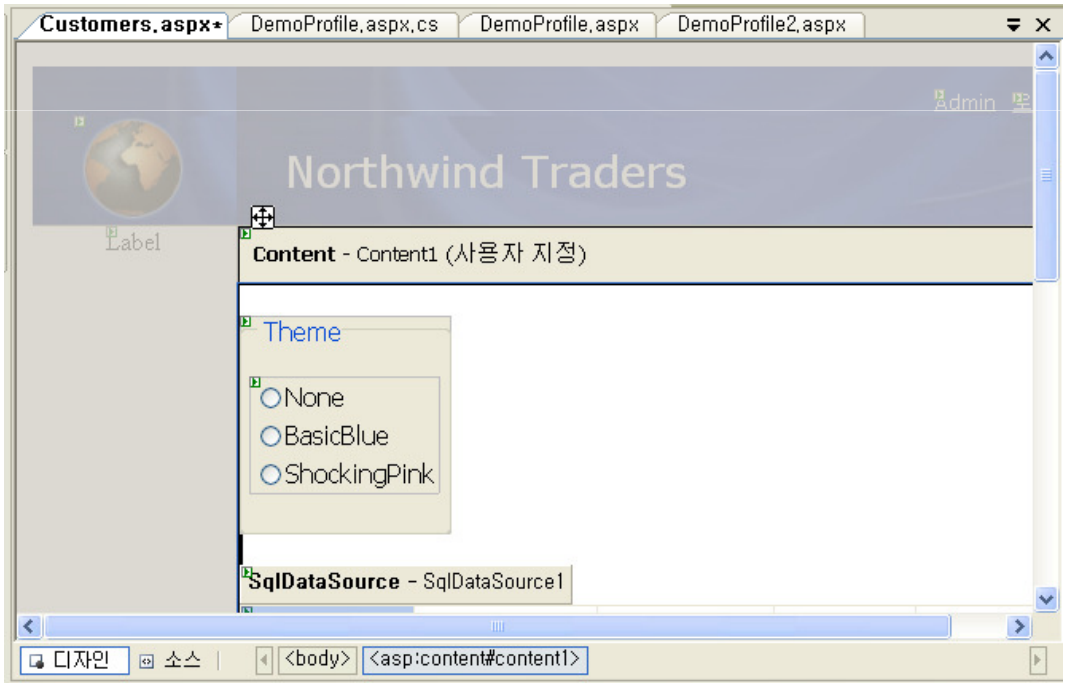
- 프로필 정의 하기
  - 솔루션 탐색기에서, web.config을 선택하여 <system.web>밑에 다음을 추가한다.

```
<anonymousIdentification enabled="true"/>
<profile>
      <properties>
        <add name="BackColor" allowAnonymous="true"/>
        <add name="Theme" type="System.String" allowAnonymous="true"/>
        <add name="LastVisit" type="System.DateTime"
         allowAnonymous="true"/>
       </properties>
</profile>
```

실습 14 : 프로필 II

- **Cutomers.aspx 페이지에 컨트롤 추가하기**
  - Customers.aspx를 디자인 창으로 열어서 SqlDataSource1컨트롤 위에 Panel을 추가하고 GroupingText속성을 “Theme”으로 지정한다
  - Panel1위에 RadioButtonList을 추가하고, 다음 속성을 지정

    - AutoPostBack
      : True
    - Items 추가
      : None
      : BasicBlue
      : ShockingPink



실습 14 : 프로필 III

- 코드에서 Profile.Theme 사용하기
  - 솔루션 탐색기에서, customers.aspx를 소스창으로 열어서 themeListID라는 static 변수를 선언한다.
  - Theme을 코드에서 적용하기 Page_PreInit()이벤트를 추가한다.

```
static string themeListID = null;

protected void Page_PreInit(object sender, EventArgs e)
{
    if (Page.IsPostBack) // Theme 선택시
    {
        if ( (string)Request[themeListID] == "None")
            Profile.Theme = null;
        else
            Profile.Theme = (string)Request[themeListID];
   }
   Page.Theme =
     (Profile.Theme=="None" || Profile.Theme=="")?null : Profile.Theme;
}
```

실습 14 : 프로필 IV

- **Page_Load()이벤트에 추가 : RadioButtonList를 초기화하기 위해**

```
protected void Page_Load(object sender, EventArgs e)
{
   if (!IsPostBack && themeListID == null) // 처음 Load시 themeListID 초기화
         themeListID = RadioButtonList1.UniqueID;

   if (!IsPostBack) // 처음 PageLoad시 Theme  표시하기
   {
           if (Page.Theme != "")
           {
               foreach (ListItem item in RadioButtonList1.Items)
               {

                   if (item.Text == Page.Theme)
                   {
                       item.Selected = true;
                       break;
                   }
               }
    }
    else  // Page.Theme == ""(처음 실행)
         RadioButtonList1.SelectedIndex = 0;// "None"을 선택
}
```

실습 14 : 프로필 V

- **프로필 정보의 테스트 하기**
  - 브라우저에서 customers.aspx을 실행한다
  - **Theme을 변경한다.**
  - 어플리케이션을 다시 시작한다.
  - 브라우저 세션이 끝나더라도 Theme이 계속 유지가 되었는지 확인한다.
  - 페이지 상단의 "로그인"을 클릭한니다.
  - 앞에서 여러분이 만들었던 계정 중 하나(Administrator)를 이용하여 로그인한다.
  - 앞 세션에서 선택했던 Theme이 변경된 것을 확인한다.
  - 브라우저를 종료한다.

실습 14 : 프로필 VI

- 익명 사용자 프로필 바꾸기
  - 솔루션 탐색기에서 "새 항목 추가"를 선택하고, "전역 응용 프로그램 클래스"를 추가한다
  - 다음 이벤트를 추가한다.

```
protected void Profile_MigrateAnonymous(object sender,
                                        ProfileMigrateEventArgs e)
{

        if (Profile.Theme  == "")
            Profile.Theme = Profile.GetProfile(e.AnonymousID).Theme;

}
```

- 테스트 하기
  - Customers.aspx를 실행해서, Theme을 선택
  - 페이지 상단의 "로그인"을 클릭
  - 앞에서 만들었던 계정 중 하나(Administrator)로 로그인 Theme이 그대로인지 확인
  - 브라우저를 닫고, Visual Studio 종료

15장 웹 파트

- 웹 파트 들어가기
- 웹 파트 영역(Web Part Zone)
- 카탈로그 영역(Catalog Zone)
- 편집기 영역(Editor Zone)

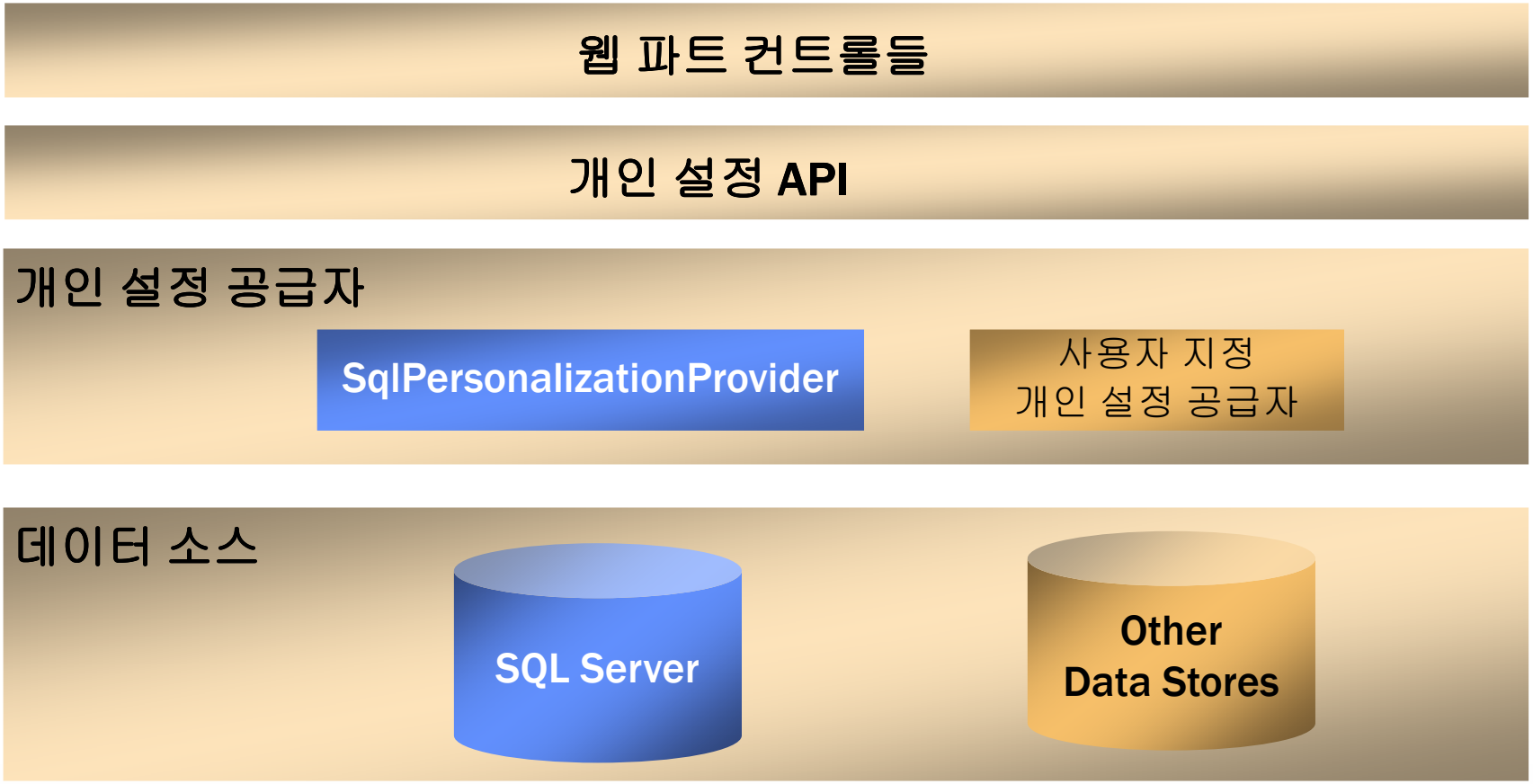
목차

Web Parts란?

- **Portal-style 어플리케이션을 개발하는 Framework**
  - SharePoint Portal Server
  - System.Web.UI.WebControls.WebParts
- **최소한의 코드로 풍부한 UI를 구현**
  - 드래그-앤-드롭으로 Page layout 를 편집
  - Appearance과 behavior 등을 편집
- **Seamless personalization**
- **Intercommunication ("connections")**

1> 웹 파트 개인 설정의 구조

- **SqlPersonalizationProvider사용**
  - C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\CONFIG\Web.config

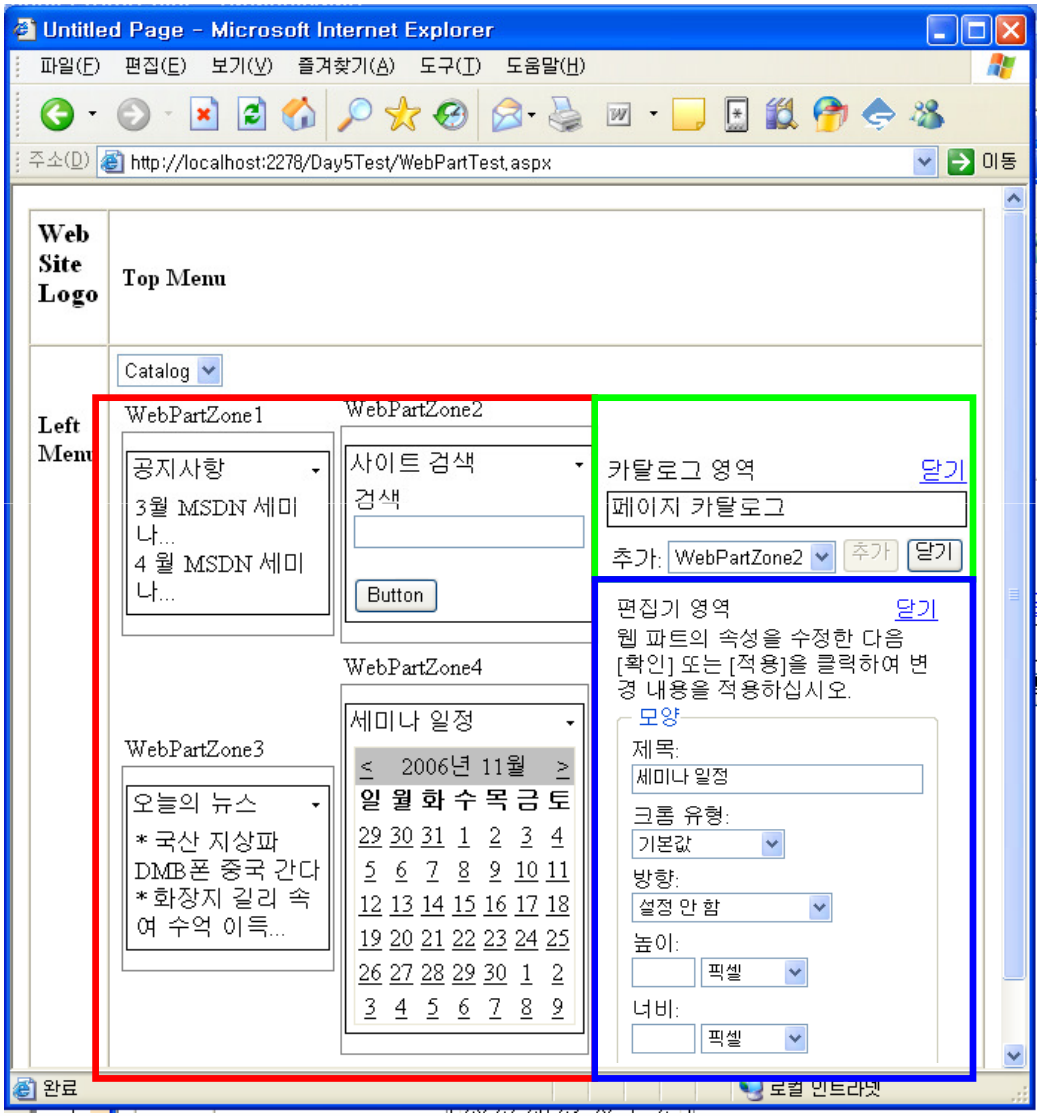


2> 웹 파트 페이지(Web Part Page)

- **웹 파트 페이지**
  - 웹 파트 컨트롤을 포함하는 일반적인 .aspx 페이지
- **다음 영역으로 분류**
  - 웹 파트 영역(Web Part Zone)
  - 카탈로그 영역(Catalog zone) : 사용자가 페이지에 Web Part를 추가할 수 있다
  - 편집기 영역(Editor Zone): 사용자가 Web Part 속성과 레이아웃을 변경할 수 있다

◆2. 웹 파트 영역(Web Part Zone)

- **WebPartZone 컨트롤**
  - WebPartManager 컨트롤
- **웹 파트 이동하기**



1> WebPartZone 컨트롤

- **WebPartZone컨트롤**
  - 웹 파트 페이지에 영역(zone)을 정의
  - 각 영역에 있는 웹 파트의 Default Layout과 모양을 정의
- **Web Part**
  - WebPartZone에 정의되는 컨트롤
    - Web controls, user controls, custom controls
  - IWebPart를 구현하지는 않았지만 GenericWebParts를 랩핑한 컨트롤
    -Title, Description 등의 속성을 추가

```
<asp:WebPartZone ID="WeatherZone"
  DragHighlightColor="244,198,96" RunAt="server">
  <PartTitleStyle BackColor="#2254B1" ForeColor="White" />
  <PartStyle BorderColor="#81AAF2" BorderStyle="Solid" BorderWidth="1px" />
  <ZoneTemplate>
        <!-- Web Parts declared here -->
    <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
       <asp:Button ID="Button1" runat="server" Text="Button" />
  </ZoneTemplate>
</asp:WebPartZone>
```

WebPartManager 컨트롤

- 웹 파트 작동을 조정한다
  - Web Parts와 Zones 리스트를 관리
  - 페이지 상태정보를 관리하고 상태가 변경되면 이벤트를 발생
  - Web Parts간의 통신을 돕는다
  - Personalization를 관리
- 페이지당 하나의 인스턴스; UI가 없다

```
<asp:WebPartManager ID="WebPartManager1" RunAt="server" />
```

2> 웹 파트 이동하기

- **WebPartManager.DisplayMode**
  - 페이지의 Display Mode를 변경하거나 얻을 수 있다
  - 웹 파트 페이지는 한 번에 하나의 디스플레이 모드만 유지할 수 있다.
  - 웹 파트 이동 가능한 모드 (인증된 사용자)
    - EditDisplayMode, DesignDisplayMode, CatalogDisplayMode

값	설명
BrowserDisplayMode	"Normal" display mode; 편집이 불가능(default)
DesignDisplayMode	드래그 앤 드롭 레이아웃 편집이 가능
EditDisplayMode	Web Part의 Appearance과 Behavior을 편집할 수 있다
CatalogDisplayMode	페이지에 Web Parts를 추가할 수 있다
ConnectDisplayMode	Web Part간에 연결 할 수 있다

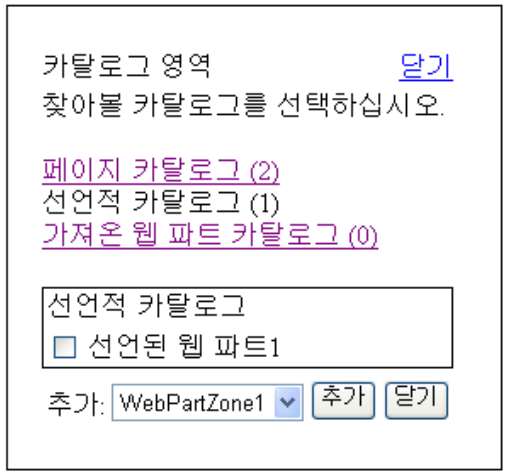
◆3. 카탈로그 영역(CatalogZone)

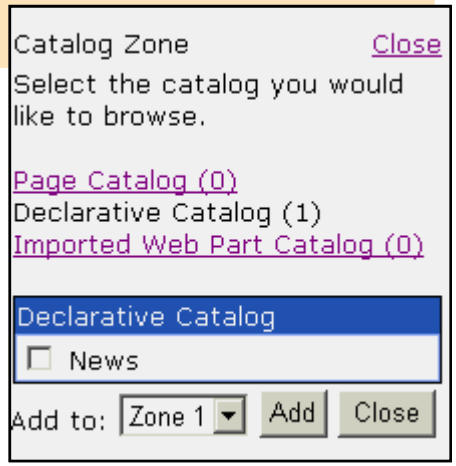
- 카탈로그 영역은 닫힌 웹 파트, 선언적인 웹 파트 및 외부에서 가져온 웹 파트를 관리해주는 영역
- **CatalogZone 컨트롤**
  - Web Parts가 대화형으로 추가될 수 있다
  - 하나 이상의 CatalogPart 컨트롤을 가진다

이름	설명
PageCatalogPart	웹 페이지에서 제거된 Web part들을 나열
DeclarativeCatalogPart	<WebPartsTemplate>에 선언된 Web Part들을 나열
ImportCatalogPart	.WebPart 파일로부터 Web Part를 가져올 수 있다

CatalogZone 선언하기


```
<asp:CatalogZone ID="CatalogZone1" Runat="server">
  <ZoneTemplate>
    <asp:PageCatalogPart ID="PageCatalogPart1" Runat="server" />
    <asp:DeclarativeCatalogPart ID="DeclarativeCatalogPart1" Runat="server">
      <WebPartsTemplate>
        <!-- Declarative Web Parts go here -->
        <asp:FileUpload ID="FileUpload1"runat="server" Title="선언된 웹 파트1"/>
      </WebPartsTemplate>
    </asp:DeclarativeCatalogPart>
    <asp:ImportCatalogPart ID="ImportCatalogPart1" Runat="server" />
  </ZoneTemplate>
</asp:CatalogZone>
```





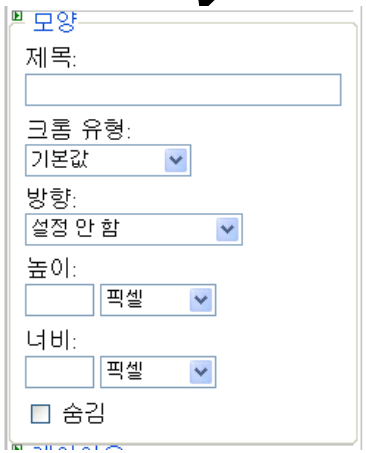
◆ 4. 편집기 영역(Editor Zone)

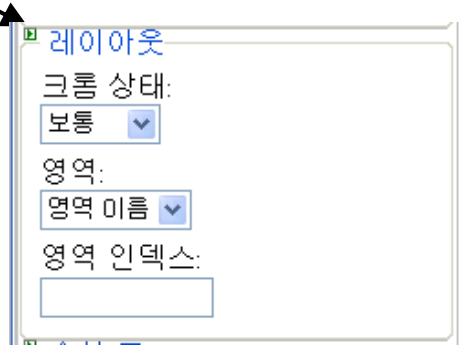
- **EditorZone 컨트롤**
  - 웹 파트의 모양, 레이아웃 및 동작 등의 설정을 관리
  - Web parts의 대화형 편집이 가능하다
  - 하나 이상의 EditorPart 컨트롤을 가진다

이름	설명
AppearanceEditorPart	타이틀과 다른 관련된 속성들을 편집할 수 있는 UI를 제공
BehaviorEditorPart	Behavior에 관련된 속성들을 편집할 수 있는 UI를 제공 (예. Web Part를 닫을 수 있는 지?)
LayoutEditorPart	Web Part의 Display 상태(minimized 또는 restored), Zone, Zone Index를 편집할 수 있는 UI를 제공
PropertyGridEditorPart	Custom 속성들을 편집할 수 있는 Property Grid를 제공

모양 편집기 파트와 레이 아웃 편집기 파트

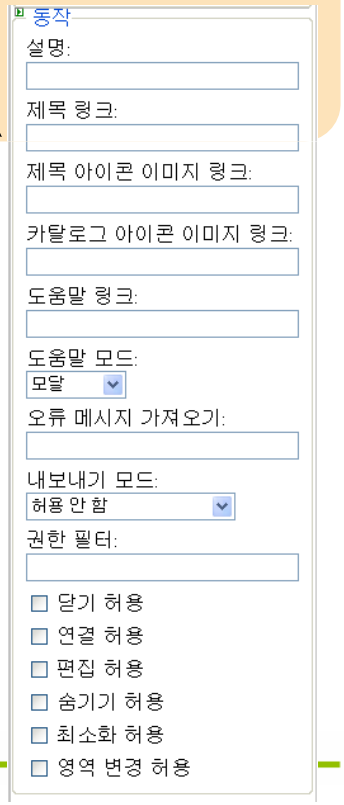

```
<asp:EditorZone ID="EditorZone1" Runat="server">
  <ZoneTemplate>
    <asp:AppearanceEditorPart ID="AppearanceEditorPart1" Runat="server" />
    <asp:LayoutEditorPart ID="LayoutEditorPart1" Runat="server" />
  </ZoneTemplate>
</asp:EditorZone>
```





동작 편집기 파트


```
<asp:EditorZone ID="EditorZone1" Runat="server">
  <ZoneTemplate>
    <asp:AppearanceEditorPart ID="AppearanceEditorPart1" Runat="server" />
    <asp:BehaviorEditorPart ID="BehaviorEditorPart1" Runat="server" />
    <asp:LayoutEditorPart ID="LayoutEditorPart1" Runat="server" />
  </ZoneTemplate>
</asp:EditorZone>
```




```
<system.web>
    <webParts enableExport="true">
      <personalization>
        <authorization>
          <allow verbs="enterSharedScope" users="admin"/>
        </authorization>
      </personalization>
    </webParts>
<system.web>
```

16장 캐싱을 이용한 성능 향상

- 캐싱 들어가기
- 캐싱 사용하기
  - 출력 캐싱(Output Caching)
  - 데이터 캐싱
- ASP.NET 캐싱의 새로운 기능
  - SQL 캐시 종속성

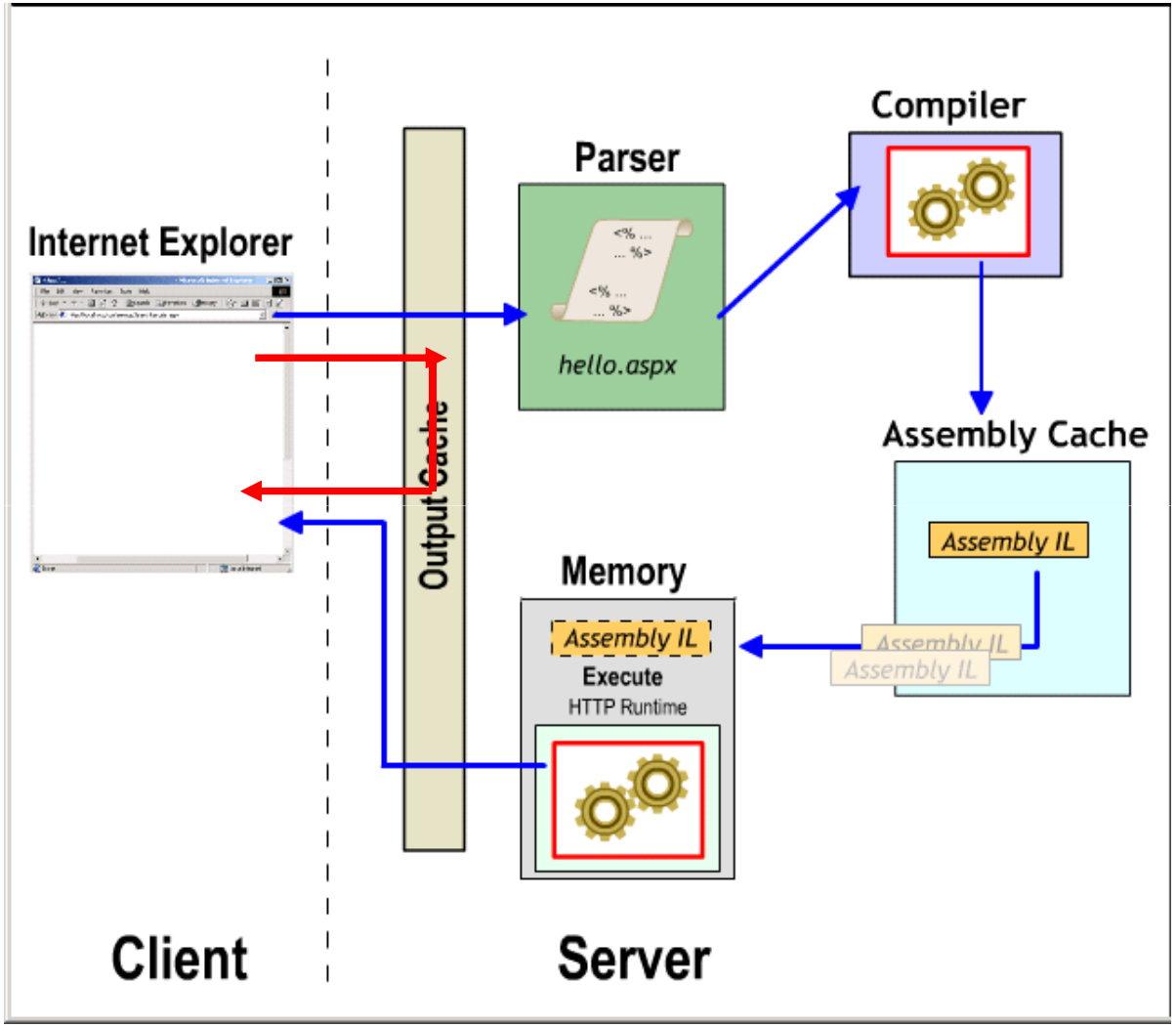
1. 캐시 들어가기

- **캐싱(Caching)이란?**
  - 캐시(Cache)란 빠른 데이터 처리를 위하여 이미 처리된 데이터를 임시적으로 저장해두는 장소
- **ASP.NET 2.0 캐싱의 종류**
  - 출력 캐싱(Output Caching)
    - -전체 페이지 캐싱
    - -부분 페이지 캐싱
      - -컨트롤 캐싱
      - -캐시 후 대체(Post-Cache Substitution)
  - 데이터 캐싱

2. 캐시 사용하기

- 출력 캐싱
- 데이터 캐싱

1>출력 캐싱(Output Caching)



전체 페이지 캐싱


```
<%@ OutputCache Duration="10" VaryByParam="none" %>
```

- **Cache content is generated from dynamic pages**
- **Entire Web page is available in cache**
- **Duration – 캐시가 유지되는 시간 (초)**
- **VaryByParam – Pos또는 Get방식으로 넘어오는 변수별로 전체 페이징을 할 때 사용하는 변수명을 설정**

부분 페이지 캐싱

- **컨트롤 캐싱**
  - 동적인 웹 부분에서 특정 부분만 정적일 때 유용

```
<%@ Control Language="C#" ClassName="WebUserControl" %>
<%@ OutputCache Duration="120" VaryByParam="none" %>
```

- **캐시 후 대체**
  - 정적인 웹 페이지에서 특정 부분만 동적일 때 유용

    -Substitution컨트롤을 사용하여 선언적으로 구현

    -Substitution API를 사용하여 프로그래밍 방식으로 구현

    -AdRotator컨트롤을 사용하여 암시적으로 구현

    : 내부적으로 이미 구현

2> 데이터 캐싱

- **An object used to store information**
  - One **Cache** object per Web Application
  - An alternative to application variables
  - Not used to store information in session variables
- **Uses key-value pairs to store and retrieve items**

```
//Implicit method
Cache["myKey"] = myValue;

//Explicit method
Cache.Insert("myKey", myValue, Dependency, AbsoluteExpiration,
 SlidingExpiration, CacheItemPriority, CacheItemRemovedCallBack);
```

```
myValue = Cache("myKey")
```

Removing Items from the Cache Object

- **AbsoluteExpiration time**

```
DateTime.Now.AddMinutes(5)
```

- **SlidingExpiration time**

```
TimeSpan.FromSeconds(20)
```

- **Dependent on a changed value**

```
AddCacheItemDependency("Variable.Value")
```

- **Cache item priority**

```
CacheItemPriority.High
```

3. ASP.NET2.0 캐싱의 새로운 기능

- 캐시 프로필
  - 응용 프로그램의 Web.config에 캐시 설정을 만든 다음 개별 페이지에서 이 설정을 참조
  - 한 번에 여러 페이지에서 캐시 설정을 적용할 수 있으므로 캐시 설정을 일괄적으로 수정해야 되는 경우 매우 유용

```
<configuration>
  <system.web>
    <caching>
        <outputCacheProfiles>
            <add name="Cache60Seconds" duration="60" varyByParam="none"/>
        </outoutCahceProfiles>
    </caching>
  </system.web>
</configuration>
```

```
<%@ OutputCache CacheProfile="Cache60Seconds" %>
```

- SQL 캐시 종속성

SQL 캐시 종속성

- 새로운 캐시 종속성 타입
  - SqlCacheDependency 클래스로 구현
  - <sqlCacheDependency>에 구성정보 명시
- 케시된 항목을 데이터베이스 엔터티와 연결
  - ASP.NET Application Cache
  - ASP.NET Output Cache
- **SQL Server 7, 2000, 2005 서로 호환됨**

데이터베이스 준비하기

- **Aspnet_regsql.exe나 SqlCacheDependencyAdmin를 사용하여 데이터베이스를 준비한다**

```
aspnet_regsql -S localhost -E -d Pubs -ed
```

서버 이름
Trusted connection
데이터베이스 이름
데이터베이스 활성화

- **Aspnet_regsql.exe나 SqlCacheDependencyAdmin를 사용하여 테이블 준비한다**

```
aspnet_regsql -S localhost -E -d Pubs -t Authors -et
```

서버 이름
Trusted connection
데이터베이스 이름
테이블 이름
테이블활성화

Web.config 수정하기


```
<configuration>
  <connectionStrings>
    <add name="pubsConnectionString"
      connectionString="server=localhost;database=pubs;..." />
  </connectionStrings>
  <system.web>
    <caching>
      <sqlCacheDependency enabled="true" pollTime="5000">
        <databases>
          <add name="PubsDB" connectionStringName="pubsConnentionString"/>
        </databases>
      </sqlCacheDependency>
    </caching>
  <system.web>
</configuration>
```

SqlCacheDependency 사용하기

- Application Cache

```
Cache.Insert (“Authors", authors,
    new SqlCacheDependency (“PubsDB", “Authors");
```

- Output Cache

```
<%@ OutputCache Duration="60" VaryByParam="None"
    SqlDependency="PubsDB:Products" %>
```

- SqlDataSource

```
<asp:SqlDataSource ID=“SqlDataSource1" RunAt="server"
  ConnectionString="server=localhost;database=pubs;..."
  SelectCommand="select distinct state from authors order by state"
  EnableCaching="true" CacheDuration="60000"
  SqlCacheDependency=“PubsDB:Authors" />
<asp:DropDownList ID="MyDropDownList" DataSourceID=“SqlDataSource1"
  DataTextField=“state" AutoPostBack="true" RunAt="server" />
```

실습 16 : SQL 캐시 종속성 -I

SQL 캐시 종속성 사용하기 전

- MasterDetials.aspx를 실행해서 GridView 확인
- SqlDataSource1의 EnableCaching 속성을 true로 변경, CacheDuration를 300으로 변경
- MasterDetails.aspx를 실행해서 첫 번째 행, Nancy Davolio 의 Title값 확인
- SQL Server Management Studio를 사용하여 Nancy Davolio의 Title를 “CEO”로 변경
- 브라우저에서 새로 고침해서 Title이 변경되지 않은 것 확인하기 (캐시 데이터가 사용되기 때문에)

실습 16: SQL 캐시 종속성 -II

SQL 캐시 종속성 사용하기 위해

- Microsoft Visual Studio 2005-> Visual Studio Tools ->Visual Studio 2005 명령 프롬프트에서 다음과 같이 데이터베이스 설정한다

  C>aspnet_regsql –S localhost –E –d Northwind –ed
  C>aspnet_regsql –S localhost –E –d Northwind –t Employees -et

- SQL Server Management Studio에서 Northwind 데이터베이스에 AspNet_SqlCacheTablesForChangeNotification 테이블이 추가된 것 확인하기
- Web.config에 다음의 <caching> 태그를 추가하고 모두 저장한다

```
<system.web>
  <caching>
    <sqlCacheDependency enabled="true" pollTime="5000">
      <databases>
        <add name="Northwind" connectionStringName="NorthwindConnectionString" />
      </databases>
    </sqlCacheDependency>
  </caching>
```

실습16: SQL 캐시 종속성 -III

- MasterDetails.aspx을 디자인 모드로 열어서 SqlDataSource1의 속성 중 SqlCacheDependency 을 속성창에서 Northwind:Employees으로 지정한다
- MasterDetials.aspx를 실행한다

SQL 캐시 종속성 확인

- SQL Server Management Studio를 사용하여 Nancy Davolio의 Title를 “Marketing Director”로 변경한다.
- 앞에서 띄웠던 브라우저에서 몇 초 기다린 후 , MasterDetials.aspx를 새로 고침해서 데이터가 변경된 것을 확인한다.