**프로필(Profiles)**

|  |
| --- |
| 싸이월드 최초 가입시 제공되는 미니홈피의 기본적인 레이아웃과 기능은 어느 회원이나 동일하지만 개인의 성향에 따라 메뉴 종류, 메뉴 위치, 배경화면, 배경음악, 캐릭터 등을 바꿀 수 있다.  이렇게 응용프로그램을 개인의 성향에 맞게 설정하는 것을 개인 설정(Personalization)이라고 한다.  개인 설정을 손쉽게 구현하기 위해 ASP.NET 2.0에서는 프로필 이라는 새로운 기능을 제공한다.  프로필을 사용하면 인증된 사용자의 설정 정보는 물론이고, 익명 사용자의 설정 정보까지 모두 저장할 수 있다. |

1. **프로필 들어가기**
2. 프로필이란?

웹 응용프로그램의 개인 설정을 위해 ASP.NET 2.0에서 제공하는 기능임

“사용자 정보를 쿠키에 저장하면 되지 않을까?”

🡺 쿠키는 보안적용이 되지 않아 불안함.

따라서 개인 설정을 위해 사용자 설정 정보는 서버측에서 안전하게 관리한다.

🡺 레이아웃이라든가 콘텐츠 위치를 변경하면 설정 정보가 해당 계정에만 영향을 미치

며 로그 아웃 후 다시 로그인하더라도 이전에 설정이 계속적으로 유지된다.

즉, 사용자 중심의 응용 프로그램이 될 수 있도록 관련 설정을 사용자가 직접 설정

하는 것을 개인 설정이라 함

1. 프로필의 구조

**프로필 API**

**프로필 공급자**

(SqlProfileProvider) (Access ProfileProvider) (사용자지정 프로필 공급자)

**데이터 소스**

SQL Server JET(Access) 사용자 지정 데이터 소스

\*) 프로필 공급자가 프로필 API와 데이터 소스 사이에 위치해 있으면서 사용자 설정 정보

를 입력하고 참조하는 프로세스를 자동으로 처리해 준다.

\*) 프로필 공급자는 기본적으로 Machine.config 파일에 정의되어 있다.

\*) 기본 프로필 공급자는 SQL Server Express, 파일 기반 DB 를 사용한다.

프로필 관련 정보는 aspnet\_Profile이라는 테이블에 저장된다.

1. **프로필 사용하기**
2. 프로필 속성(Profile Property) 선언

프로필을 사용하여 사용자 설정 정보를 저장하기 위해 구성 파일에 속성을 선언해야 함

🡺 해당 속성은 프로필에서 사용자 설정 정보를 저장하고 참조하는 데 사용되는 속성임

🡺 Web.config 파일에 속성을 선언하게 된다.

|  |
| --- |
| <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> |

* 4개의 프로필 속성을 선언(add name의 MainBackColor 외…)
* 실제 사용자 설정 정보는 4개의 프로필 속성을 통해 데이터 베이스에 저장되고 참조된다.
* 속성에는 저장형식을 지정할 수 있고 , defult는 System.String 형식을 가지게 된다.
* 프로필 속성은 그룹화 할 수 있다.
* 이렇게 선언된 프로필 속성은 웹 응용 프로그램의 모든 페이지에서 아래와 같이 사용할 수 있게 된다.

|  |
| --- |
| // 사용자 설정 정보 입력  Proflie. MainBackColor = “red”;  Profile. EnableCalendar = true;  Profile.MemInfo.NickName = “동네 노는 청년”;  Profile.MemInfo.Age = 29;  // 사용자 설정 정보 참조  String mainBackColor = Proflie. MainBackColor;  bool enableCalender = Profile. EnableCalendar;  string nickname = Profile.MemInfo.NickName;  int age = Profile.MemInfo.Age; |

1. 인증된 사용자를 위한 프로필

구성 파일에 선언된 속성은 기본적으로 인증된 사용자를 위한 프로필 속성이 된다.

익명 사용자를 위한 속성은 구성 파일에서 몇 가지 설정이 더 필요하다.

예제 확인)

1. 익명 사용자를 위한 프로필

익명 사용자를 위한 프로필을 사용하기 위해서는 구성파일에<anonymousIdentificaton>요소를 추가하고 이 요소의 enable 특성을 true로 지정한 후, 각 프로필 속성 선언시 다음과 같이 allowAnonymous 특성을 true로 지정해야 한다.

|  |
| --- |
| <anonymousIdentification enabled="true"/>  <profile >  <providers>  <add name="MainBackColor" allowAnonymous = “true” />  <add name="EnableCalendar" type="System.Boolean" allowAnonymous =  “true”/>  </properties>  </profile>  < !--  <location .. />  - - > |

* 익명 사용자를 위해 선언된 프로필 속성은 여전히 인증된 사용자에게도 적용된다.
* 인증도 받지 않은 익명 사용자의 설정 정보는 어디에 보관되나?
* 웹 서버는 수많은 익명 사용자들을 어떻게 구분하나?

웹 서버는 프로필 공급자를 통해 수많은 익명 사용자들을 구분할 수 있는 ID를 자체적으로 생성하며, 각 익명 사용자의 쿠키에 이 ID를 저장한다. 또한 이 ID는 서버측 데이터 베이스의 aspnet\_Users 테이블에 저장되고, 관련 프로필 정보는 aspnet\_Profile에 저장된다.

따라서 익명 사용자가 서버측에 프로필 정보를 요청해 올 경우 프로필 공급자는 익명 사용자의 쿠키를 검사해서 ID를 가져오고, 다시 이 ID를 통해 해당 프로필 정보를 aspnet\_Profile 테이블에서 가져오게 된다.

* 만약, 익명 사용자가 자신의 쿠키를 삭제하면?

서버측에 저장된 사용자 설정 정보는 그대로 남아 있지만, 익명 사용자의 쿠키에서 해당 ID를 찾을 수 없기 때문에 결과적으로 사용자 설정 정보는 초기화 된다.

웹 브라우저 [도구] / [ 인터넷 옵션 ] / [일반] / [쿠키 삭제]

* 시간이 지나면 데이터베이스에 불필요한 익명 사용자의 정보가 쌓이게 됨

이를 해결하기 위해 프로필 API가 제공됨

1. 프로필 이동( 익명 사용자 🡺 인증된 사용자)

익명 사용자가 설정한 사용자 설정 정보는 익명 사용자가 로그인한 직후 인증된 사용자가 되었을 때 인증된 사용자의 설정 정보로 이동할 수 있다.

이러한 기능은 쇼핑몰 구현 시 유용하게 사용될 수 있다.

Ex) 쇼핑몰에서 익명 사용자도 장바구니에 물품을 담아 쇼핑을 할 수 있게 만들고(익명 사용자 프로필) 그 사용자가 로그인하는 동시에 익명 사용자 프로필에 있는 물품 정보들을 인증된 사용자의 프로필로 바로 이동시켜 인증된 사용자의 장바구니를 즉시 업데이트하는 프로세스를 구현할 수 있게 된다.

* Global.asax 파일에서 처리하게 된다.
* 새항목 추가 대화상자에서 ‘전역 응용 프로그램 클래스’ 템플릿을 선택하여 추가하게 됨

Sample 확인)

1. 프로필 API

익명 사용자들 중 활동하지 않는 시간이 1년 이상 된 사용자들의 프로필 정보를 모두 삭제하는 예제

|  |
| --- |
| <%@ Page Language="C#" %>  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  <script runat="server">  protected void Page\_Load(object sender, EventArgs e)  {  if (!Page.IsPostBack)  {  BindDropDownList1();  }  }  void BindDropDownList1()  {  DropDownList1.Items.Clear();  ProfileInfoCollection pc = ProfileManager.GetAllProfiles(ProfileAuthenticationOption.All);  foreach (ProfileInfo p in pc)  {  ListItem li = new ListItem(p.UserName);  DropDownList1.Items.Add(li);  }  }    protected void Button1\_Click(object sender, EventArgs e)  {  int year = int.Parse(TextBox1.Text.Substring(0, 4));  int month = int.Parse(TextBox1.Text.Substring(4, 2));  int day = int.Parse(TextBox1.Text.Substring(6, 2));  DateTime date = new DateTime(year, month, day);  ProfileManager.DeleteInactiveProfiles(ProfileAuthenticationOption.All, date);  BindDropDownList1();  }  protected void Button2\_Click(object sender, EventArgs e)  {  if (DropDownList1.Items.Count > 0)  {  ProfileManager.DeleteProfile(DropDownList1.SelectedItem.Text);  BindDropDownList1();  }  }  </script>  <html xmlns="http://www.w3.org/1999/xhtml" >  <head id="Head1" runat="server">  <title>제목 없음</title>  </head>  <body>  <form id="form1" runat="server">  <div>  날짜 : <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox> (예: 20060101)  <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server" ErrorMessage="날짜를 입력하세요." ControlToValidate="TextBox1" Display="Dynamic" ValidationGroup="Date"></asp:RequiredFieldValidator>  <asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server" ErrorMessage="날짜 형식이 잘못되었습니다." ValidationExpression="\d{8}" ControlToValidate="TextBox1" SetFocusOnError="true" Display="Dynamic" ValidationGroup="Date"></asp:RegularExpressionValidator><br />  <asp:Button ID="Button1" runat="server" Text="입력한 날짜 이전의 모든 프로필 삭제" OnClick="Button1\_Click" ValidationGroup="Date" /><p></p>  사용자 : <asp:DropDownList ID="DropDownList1" runat="server">  </asp:DropDownList>  <asp:Button ID="Button2" runat="server" Text="선택 프로필 삭제" OnClick="Button2\_Click" /><br />  </div>  </form>  </body>  </html> |

\*) ProfileManager 클래스는 콘솔 또는 Windows 응용 프로그램에서 사용할 수 있으므로

불필요한 프로필을 삭제하는 프로세스를 콘솔 또는 윈폼으로 구현하여 [시작] / [제어판]/[예약된 작업]에 걸어두면 프로필 관리를 자동화 할 수 있다.