1.需引用Microsoft.Owin.Host.SystemWeb、Microsoft.Owin.Security.Cookies、Microsoft.AspNet.Identity.EntityFramework。

2.資料表設計，分別建立5個類別繼承於IdentityRole、IdentityUserRole、IdentityUser、IdentityUserLogin、IdentityUserClaim

，也可不繼承，程式將會直接引用所以繼承以上類別的所有屬性包括以上5個類別本身之屬性。

繼承範例:

public class AppUser : IdentityUser

{

public string Country { get; set; }

public int Age { get; set; }

public bool Sex { get; set; }

public int? SexCount { get; set; }

}

public class AppUserS : IdentityUser{

public string CountryS { get; set; }

}

以上Country 與 CountryS 將一同新增。

程式將會於Idendity相關之資料表擴充以上四個屬性，即使有另外一個類別繼承IdentityUser，其屬性也將一併納入新增範圍並不限定任何命名空間。

3.DBContext建立

注意：dbcontext的連線字串需使用資料庫之連線字串，而不能使用EntityFramework的

資料類別建立完之後即開始建立資料連線相關設置，

範例:

public class AppContext :IdentityDbContext<AppUser>

{

//如果使用者使用無參數之建構函式時，引用父類別之建構函式，兩個參數分別為 連線字串名稱、

public AppContext() : base("DefaultConnection", false) { }

public static AppContext Create()

{

return new AppContext();

}

//在建立實體資料表時的可以插手的方法

protected override void OnModelCreating(DbModelBuilder modelBuilder)

{

if (modelBuilder == null)

{

throw new ArgumentOutOfRangeException("modelBuilder is null");

}

base.OnModelCreating(modelBuilder);

//下面方法為IdentityUserRole之資料表更改其資料表名稱的範例，而Entity內的參數如果不是以上五個資料表時，將會自行建立一個全新的資料表

modelBuilder.Entity<IdentityUserRole>().ToTable("UserRole");

}

}

4.需建立Startup類別並置放於專案底下，其所需之程式碼為

//建立一個以 IdentityUser 為參數的UserManager的工廠委派函數

//註1 AppUser: IdentityUser

//註2 Func<T>內只有一個泛型時 T為 回傳類型 ，Func<TValue,TResult> 而有兩個泛型，第一個為傳入的參數，第二個為回傳類型

public static Func<UserManager<AppUser>> UserManagerFactory { get; private set; }

public void Configuration(IAppBuilder app)

{

//網站的使用者狀態管理

app.UseCookieAuthentication(new CookieAuthenticationOptions()

{

//儲存的COOKIE類型

AuthenticationType = DefaultAuthenticationTypes.ApplicationCookie,

//無登入將會引導的畫面

LoginPath = new PathString("/auth/login")

});

//使用上面所宣告之委派，建立其方法實體

UserManagerFactory = () =>

{

//usermanager = UserManager<AppUser>

//註AppContext() 為上面繼承:IdentityDbContext<AppUser> 之類別

var usermanager = new UserManager<AppUser>(new UserStore<AppUser>(new AppContext()));

//使用者的限制

usermanager.UserValidator = new UserValidator<AppUser>(usermanager) {

//使用者名稱不只能輸入英文與數字

AllowOnlyAlphanumericUserNames = false

};

return usermanager;

};

}

5.於管理器主控台依順序輸入 : Enable-Migrations(開啟產生codefirst文件功能) 、Add-Migration 文件名稱(建立該codefirst文件)、Update-Database(更新資料庫);

6.註冊功能引用:

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Register(RegisterModel model)

{

if (!ModelState.IsValid)

{

return View(model);

}

//建立一個新User

var user = new AppUser

{

UserName = model.Email,

Country = model.Country,

Age = model.Age,

};

//UserManaget儲存User與密碼進資料庫

var result = await usermanager.CreateAsync(user, model.Password);

//如果儲存成功

if (result.Succeeded)

{

//將該USER登入

await SignIn(user);

return RedirectToAction("Index", "Home");

}

foreach(var err in result.Errors)

{

ModelState.AddModelError("", err);

}

return View(model);

}

private async Task SignIn(AppUser user)

{

var identity = await usermanager.CreateIdentityAsync(user, DefaultAuthenticationTypes.ApplicationCookie);

identity.AddClaim(new Claim(ClaimTypes.Country, user.Country));

GetAuthenticationManager().SignIn(identity);

}

private IAuthenticationManager GetAuthenticationManager()

{

var ctx = Request.GetOwinContext();

return ctx.Authentication;

}

private string GetRedirectUrl(string returnUrl)

{

if (string.IsNullOrEmpty(returnUrl) || !Url.IsLocalUrl(returnUrl))

{

returnUrl = Url.Action("index", "Home");

}

return returnUrl;

}

}

7.登入功能引用:

[HttpPost]

public async Task<ActionResult> Login(UserModel model)

{

if (!ModelState.IsValid)

{

return View(model);

}

var user = await usermanager.FindAsync(model.Email, model.Password);

if (user != null)

{

await SignIn(user);

return Redirect(GetRedirectUrl(model.ReturnUrl));

}

// user authN failed

ModelState.AddModelError("", "Invalid email or password");

return View();

}