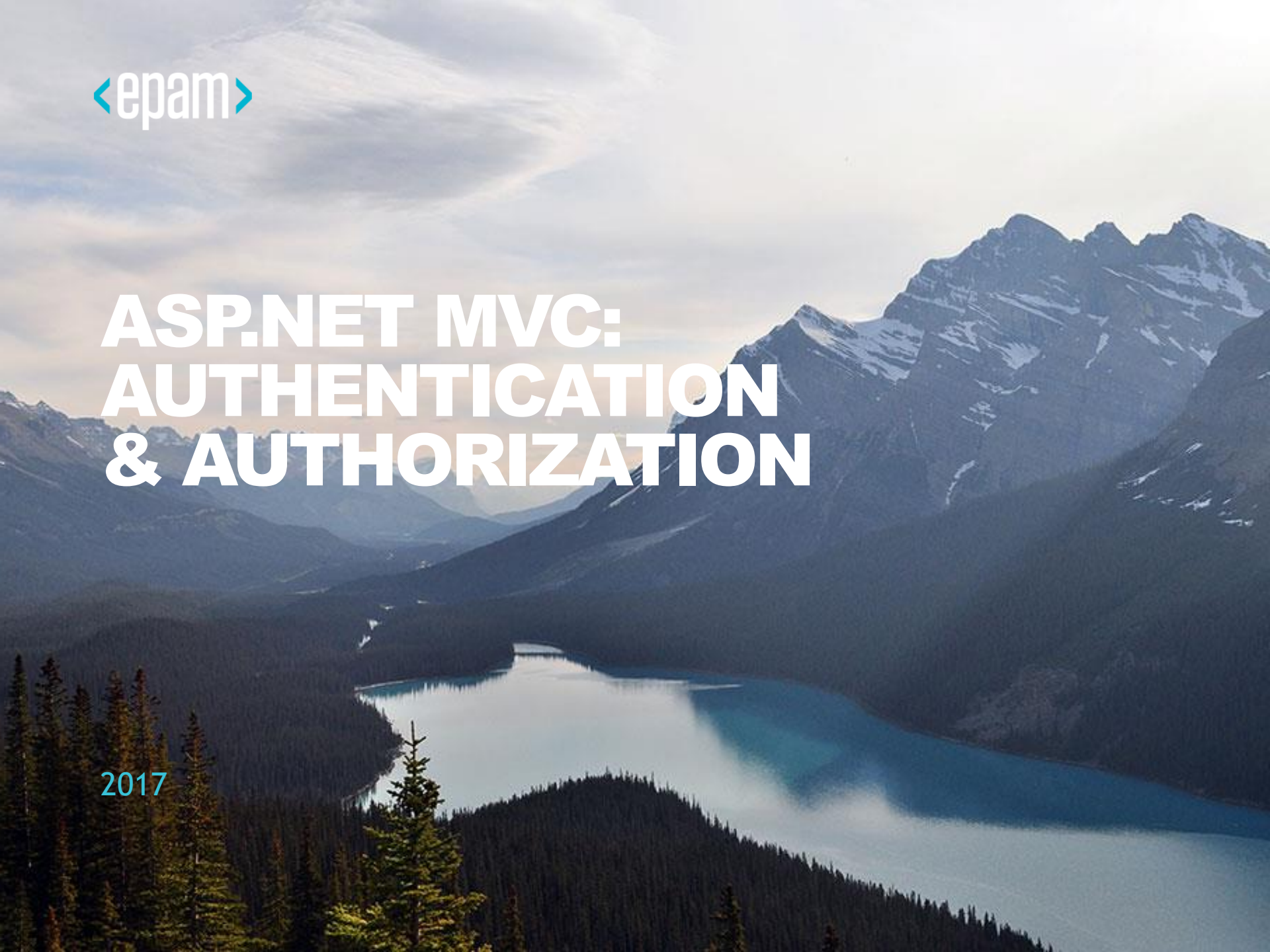

epam
ASP.NET MVC:
AUTHENTICATION
& AUTHORIZATION
2017

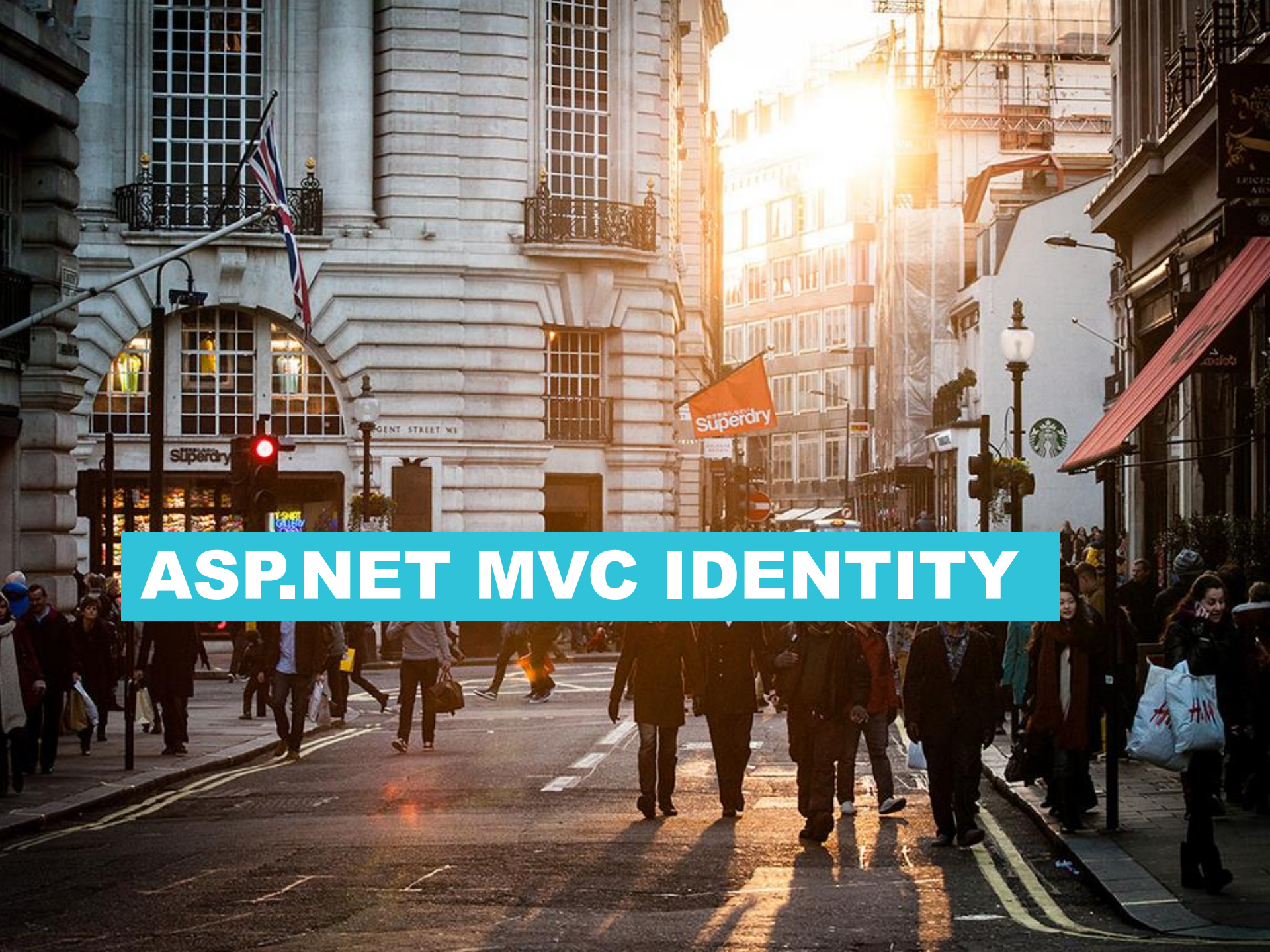
AUTHENTICATION & AUTHORIZATION

AUTHENTICATION AND AUTHORIZATION

What's the Difference?

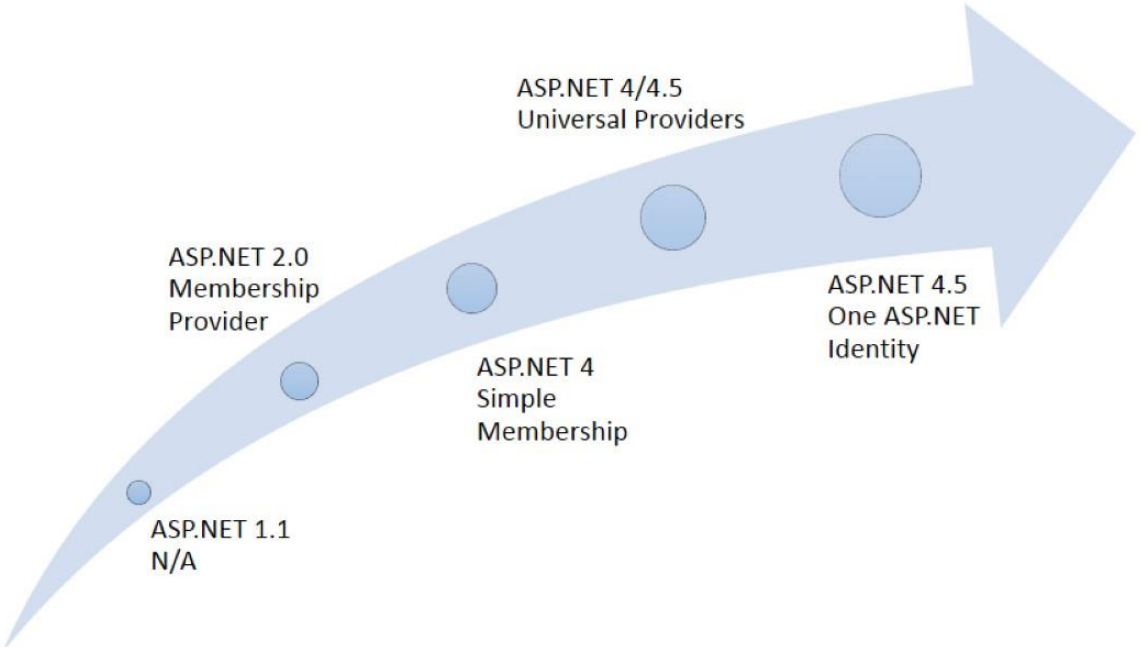
AUTHENTICATION & AUTHORIZATION

- **Authentication**
  - Процесс подтверждения личности пользователя или устройства
  - Вопросы: **Кто вы? Как вы это докажете?**
  - Личность можно подтвердить разными способами:
    - Логин и пароль
    - Ключ-карта
    - Внешний токен
- **Authorization**
  - Процесс подтверждения прав и полномочий, доступных аутентифицированному пользователю или устройству
  - Вопросы: **Что вам доступно? Можете ли вы открыть эту страницу?**


ASP.NET MVC IDENTITY

ASP.NET AUTH HISTORY

From Membership to Identity



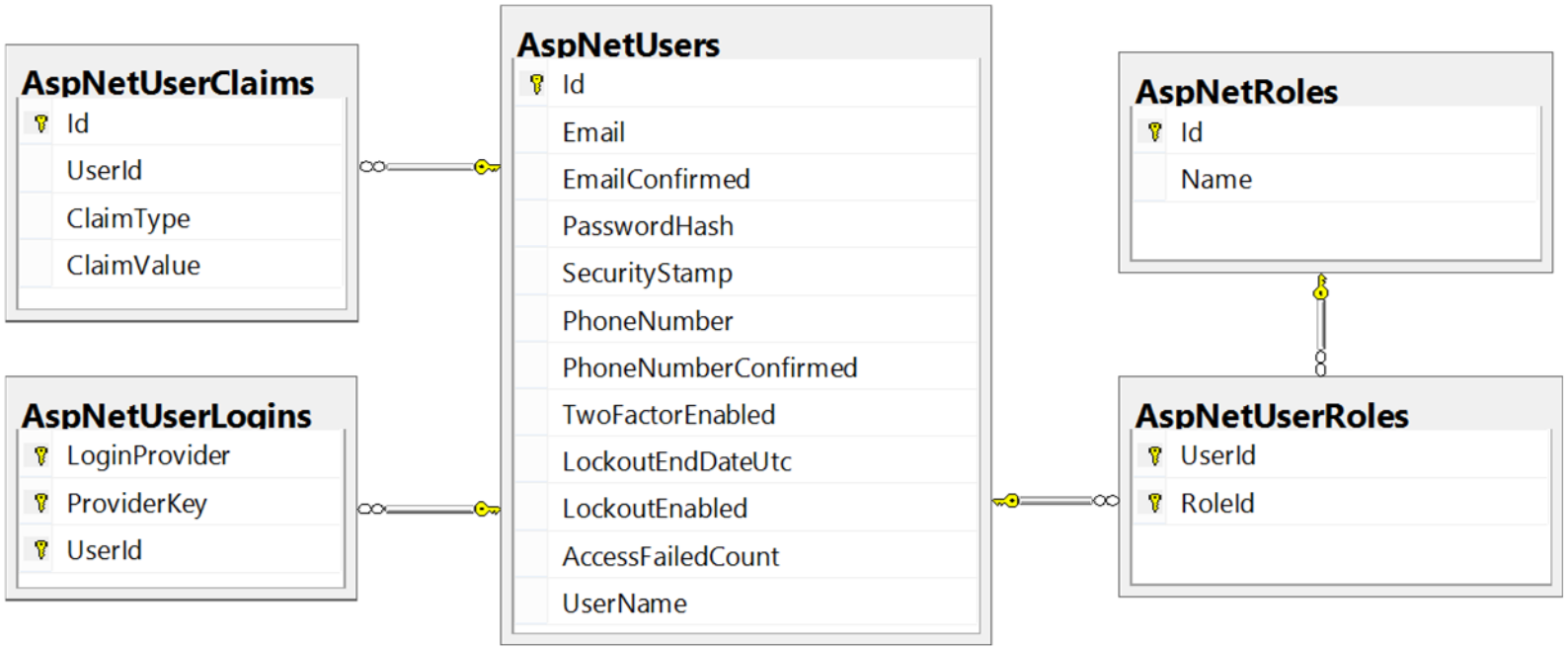
ASP.NET IDENTITY

- Система аутентификации и авторизации для всех ASP.NET сервисов и приложений
  - Поддерживает: **ASP.NET MVC, Web Api, Web Forms, SignalR, Web Pages**
- Позволяет работать с:
  - пользователями и профилями
  - внутренними и внешними логинами
  - ролями и клеймами
- Нет никакой привязки к механизму хранения данных
  - Позволяет использовать локальные и внешние хранилища
- Поддерживает сторонние системы аутентификации (OAuth)
  - «Из коробки» доступна аутентификация через учетные записи:
    - Facebook
    - Google
    - Microsoft
    - Twitter

- Основывается на **OWIN middleware** (нет привязки к **IIS**)
- Поставляется как **NuGet** пакет (частые обновления и короткие релизные циклы)

ASP.NET IDENTITY & ENTITY FRAMEWORK

Microsoft предоставляет готовую реализацию всех компонентов **ASP.NET Identity** основываясь на **Entity Framework Code First:**



ASP.NET IDENTITY SETUP

Основные способы конфигурирования ASP.NET Identity в вашем приложении:

- Создание проекта с использованием базового шаблона авторизации «**Individual User Accounts**»
- Вручную, с использованием **`Identity.EntityFramework`**:
    - Установка NuGet-пакетов
    - Конфигурирование
    - Создание моделей, контроллеров и т.д.
- Вручную, без использования **`Identity.EntityFramework`**:
    - Установка пакета **`Identity.Core`**
    - Реализация базовых компонентов системы
    - Создание моделей, контроллеров и т.д.
- Необходимые NuGet-пакеты:
    - **`Microsoft.AspNet.Identity.Core`**
    - **`Microsoft.AspNet.Identity.Owin`**
    - **`Microsoft.AspNet.Identity.EntityFramework`**

ASP.NET PROJECT TEMPLATE AUTHENTICATION

- **IdentityConfig.cs** – содержит конфигурацию менеджера пользователей
- **ApplicationUserManager : UserManager<ApplicationUser>**
  - Основной класс для управления пользователями
  - Содержит настройки валидации пользователей и паролей
- **ApplicationSignInManager : SignInManager**
  - Реализует процесс аутентификации (login/logout)
  - Поддерживает стороннюю аутентификацию
  - Поддерживает механизм двухфакторной аутентификации (email confirmation)

ASP.NET PROJECT TEMPLATE AUTHENTICATION

- **IdentityModels.cs** – содержит модель пользователя и контекст базы данных

- **ApplicationUser : IdentityUser**
  - Содержит основную информацию о пользователе в системе
  - **Id** (**уникальный** идентификатор, строка содержащая **GUID**)
    - Например, **313c241a-29ed-4398-b185-9a143bbd03ef**
  - **UserName** (**уникальное** имя пользователя), например, **Nag1bat0r**
  - **Email** (уникальность конфигурируема), например, **vasya.2005@gmail.com**
  - Также может содержать дополнительные поля, например:
    - Фамилия
    - Дата рождения
    - Ссылка на аватар

ASP.NET PROJECT TEMPLATE AUTHENTICATION

- `ApplicationDbContext : IdentityDbContext<ApplicationUser>`
  - Базовые сущности
  - Конфигурация модели
  - Инициализатор
  - и т.д.

- `Startup.Auth.cs`
  - Конфигурация использования **Identity** для **OWIN**
  - Обычно, включает аутентификацию через **cookie**
  - Может включать «внешнюю» аутентификацию

USER REGISTRATION

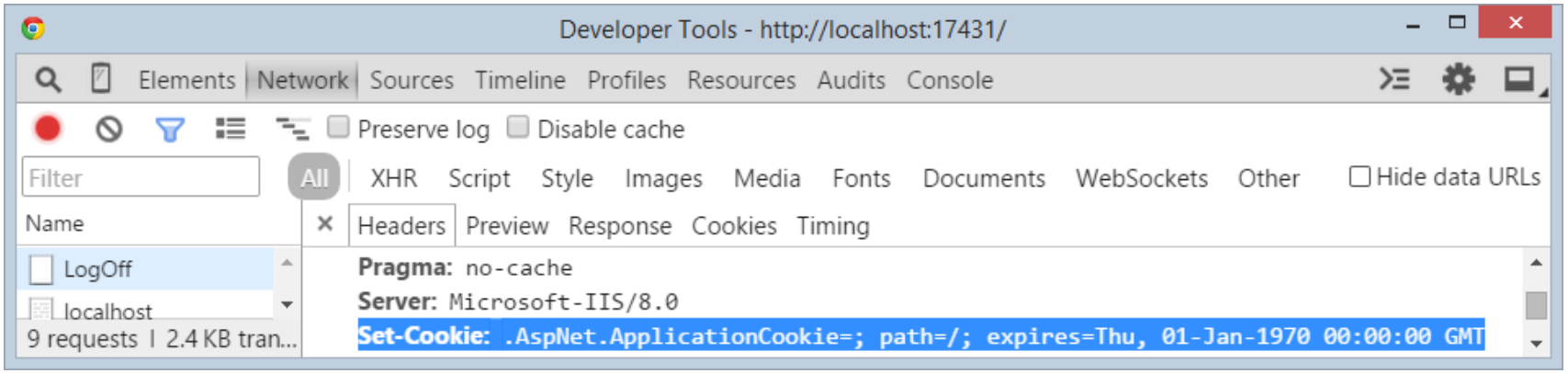
```
var newUser = new ApplicationUser
{
    UserName = "Nag1bat0r",
    Email = "vasya.2005@gmail.com",
    PhoneNumber = "+123 45 678 9101"
};

var userManager = HttpContext.GetOwinContext()
    .GetUserManager<ApplicationUserManager>();
var result = userManager.Create(newUser, "S0m3@Pa$$");

if (result.Succeeded)
    // User registered
else
    // result.Errors holds the error messages
```

USER AUTHENTICATION

```
var signInManager = HttpContext.GetOwinContext()
    .Get<ApplicationSignInManager>();

bool rememberMe = true;
bool shouldLockout = false;

var signInStatus = signInManager.PasswordSignIn(
    "Nag1bat0r", "S0m3@Pa$$", rememberMe, shouldLockout);

if (signInStatus == SignInStatus.Success)
    // Successful login
else
    // Login failed
```

USER LOGOUT

```
public ActionResult LogOut() {
{
    var authenticationManager =
        HttpContext.GetOwinContext().Authentication;

    authenticationManager.SignOut();
    // Redirect to home screen or login page
}
```



CHANGE PASSWORD

```
// Аутентифицированный пользователь меняет свой пароль
var currentUserId = User.Identity.GetUserId();
var userManager = HttpContext.GetOwinContext()
    .GetUserManager<ApplicationUserManager>();

var result = userManager.ChangePassword(
    currentUserId, "old pass", "new pass");

if (result.Succeeded) { ... }

// Администратор сбрасывает пароль у одного из пользователей
string token = userManager.GeneratePasswordResetToken(userId);
var result = userManager.ResetPassword(
    userId, token, "new pass");
```

EXTENDING THE USER PROFILE

- Для расширения профиля пользователя:
  - Добавьте свойства в класс `ApplicationUser`:

```
public class ApplicationUser : IdentityUser
{
  [Required]
  public string Name { get; set; }
  ...
}
```

  - Включите миграции / добавьте инициализатор

ASP.NET AUTHORIZATION

Используйте атрибуты [Authorize] и [AllowAnonymous] для разграничения доступа:

```
[Authorize]
public class AccountController : Controller
{
    // Get : /Account/Login (anonymous)
    [AllowAnonymous]
    public ActionResult Login(string returnUrl) { ... }

    // Post: /Account/LogOff (for logged-in users only)
    [HttpPost]
    public ActionResult LogOff() { ... }
}
```

GET USER INFORMATION

```
// GET: /Account/Roles (for logged-in users only)
[Authorize]
public ActionResult Roles()
{
    var currentUserId = User.Identity.GetUserId();
    var userManager = HttpContext.GetOwinContext()
        .GetUserManager<ApplicationUserManager>();

    var user = userManager.FindById(currentUserId);
    ViewBag.Roles = user.Roles;

    return View();
}
```

CREATE NEW ROLE

- Роли представляют из себя группы пользователей и служат для упрощения разграничения доступа.
    - Котроллеры и action-ны способны фильтровать доступ в зависимости от наличия конкретный ролей у пользователя.

```
var roleManager = new RoleManager<IdentityRole>(
    new RoleStore<IdentityRole>(new AppliationDbContext()));

var roleCreateResult = roleManager.Create(
    new IdentityRole("Administrator"));

if (!roleCreateReult.Succeeded)
{
    throw new Exception(string.Join("; ", roleCreateResult.Errors));
}
```

ADD USER TO ROLE

```
var userManager = HttpContext.GetOwinContext()
    .GetUserManager<ApplicationUserManager>();

var addAdminRoleResult = userManager.AddToRole(
    adminUserId, "Administrator");

if (addAdminRoleResult.Succeeded)
{
    // The user is now administrator
}
```

REQUIRE LOGGED-IN USER IN CERTAIN ROLE

- Разрешить доступ только для пользователей, имеющих роль «Администратор»:

```
[Authorize(Roles="Administrator")]
public class AdminController : Controller
{ ... }
```

- Разрешить доступ для пользователей имеющих роль «User», «Student» или «Trainer»:

```
[Authorize(Roles="User, Student, Trainer")]
public ActionResult Roles()
{ ... }
```

CHECK USER IS IN ROLE

```
// GET : /Home/Admin (for logged-in admins only)
[Authorize]
public ActionResult Admin()
{
    if (User.IsInRole("Administrator"))
    {
        ViewBag.Message = "Welcome to the admin area!";
        return View();
    }

    return View("Unauthorized");
}
```


CLAIMS
Superdry
Superdry

CLAIMS-BASED AUTHENTICATION

- Клеймы
  - Маленькие кусочки информации, описывающие пользователя
  - Хранятся и передаются как пары ключ-значение
  - Содержат authentication token или цифровую подпись

- Claims-based authentication
  - Пользователь аутентифицируется во внешней системе
  - Информация о пользователе передается в ваше приложение
  - Пользователь аутентифицирован и опознан

CLAIMS-BASED AUTHENTICATION

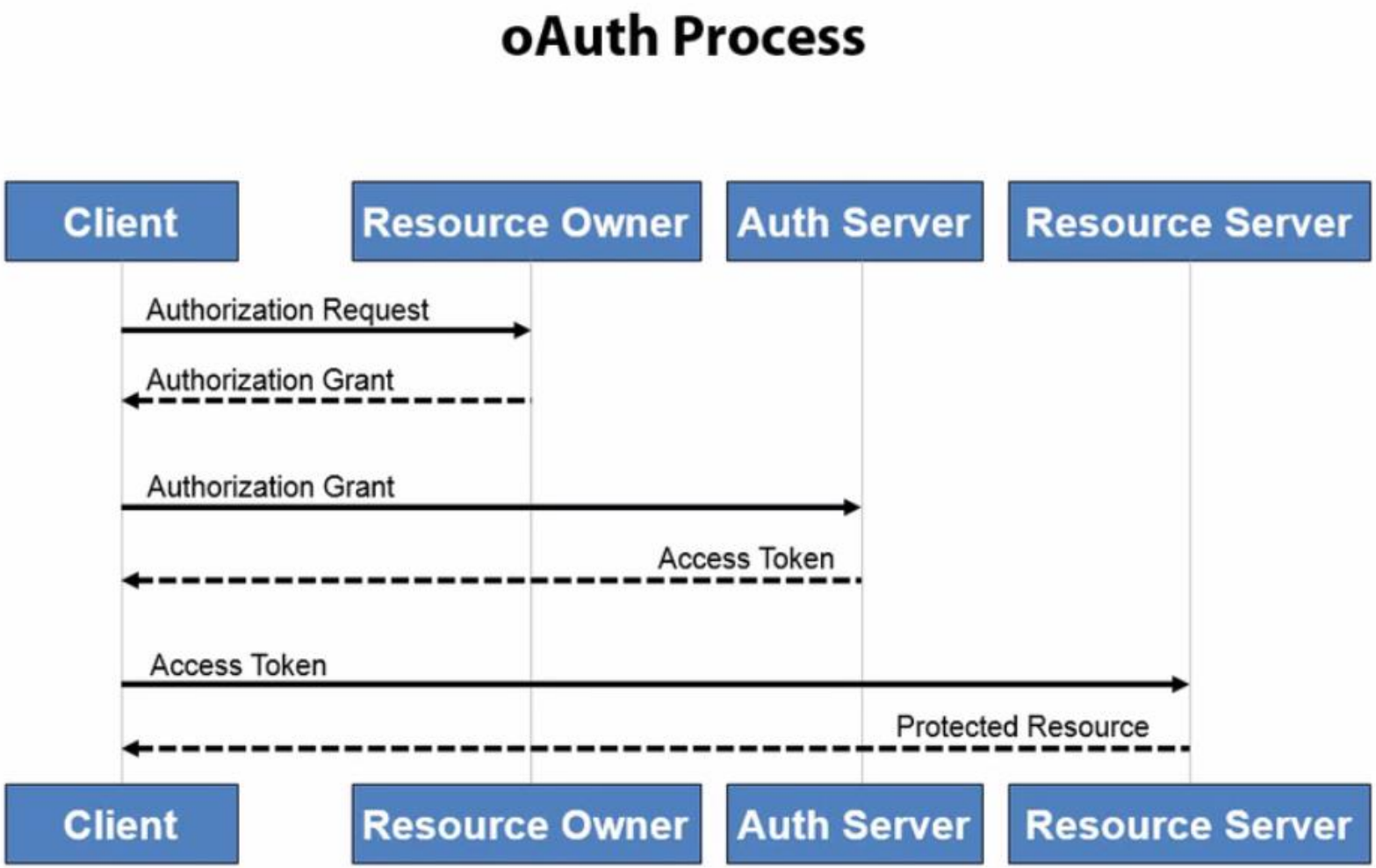
- Authentication flow
  - Пользователь выполняет запрос к вашему приложению
  - Приложение перенаправляет пользователя на стороннюю систему
  - После аутентификации сторонняя система перенаправляет пользователя обратно, но у же с информацией об этом пользователе от сторонней системы
  - Приложение делает запрос на стороннюю систему для валидации пользователя
  - Пользователь получает доступ к приложению

OAUTH2

- Позволяет выполнять безопасную аутентификацию
- Простой и стандартизированный протокол
- Может быть использован в вебе, десктопных и мобильных приложениях

- Authentication flow
    - Пользователь делает запрос к вашему приложению
    - Приложение перенаправляет пользователя к стороннему сервису
    - Сторонний сервис возвращает access token, который передается пользователю
    - Пользователь получает доступ к вашему приложению

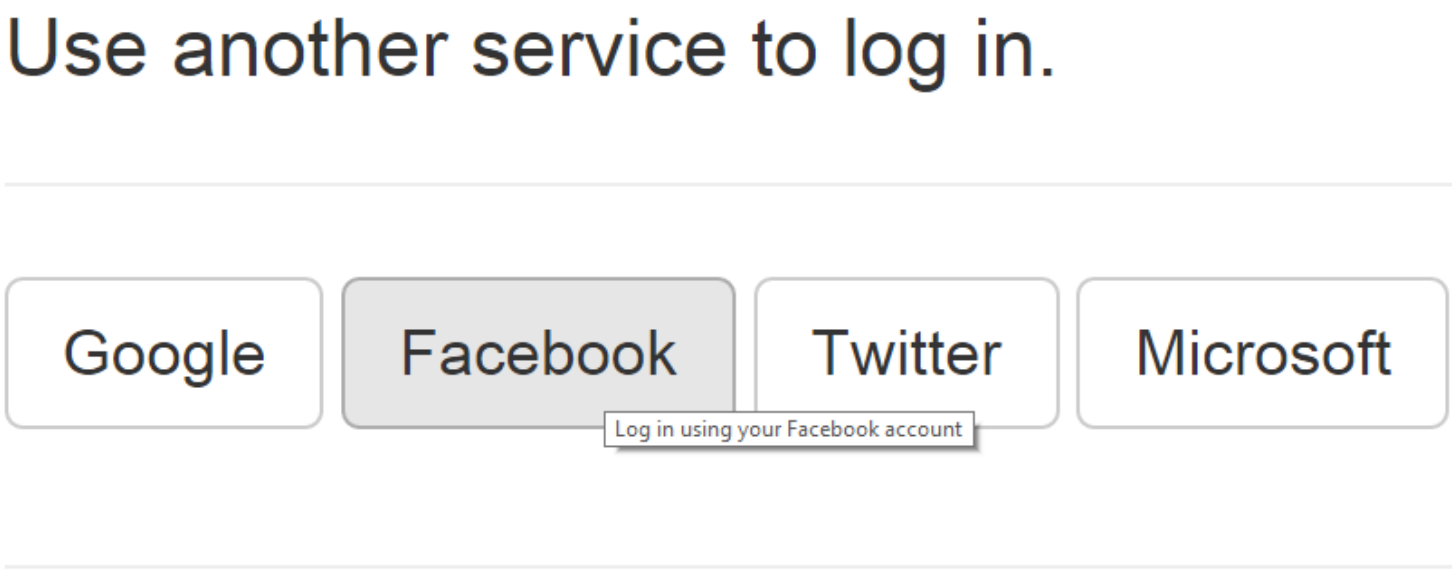
OAUTH PROCESS



ENABLE EXTERNAL LOGIN IN ASP.NET MVC

```
public partial class Startup
{
    public void ConfigureAuth(IAppBuilder app)
    {
        ...

        app.UseFacebookAuthentication(appId:"xxx", appSecret:"yyy");

        app.UseGoogleAuthentication(new GoogleOauth2AuthenticationOptions
        {
            ClientId = "xxx",
            ClientSecret = "yyy"
        });
    }
}
```

EXTERNAL LOGIN IN ASP.NET MVC



ANY QUESTIONS?

THANKS FOR YOUR ATTENTION!