Authenticating and Authorizing Users with ASP.NET Identity

Gill Cleeren
ARCHITECT

@gillcleeren www.snowball.be

Overview

User management

Extending the IdentityUser

Role management

- Role-based authorization
- Claims-based authorization
- Policy-based authorization

Adding third-party authentication

User Management

A question from Bethany

- I need to be able to manage my users from the site. Is it possible to add this to my site?

ASP.NET Core Identity

Authentication and authorization

User management built-in

- Enumerating users
- CRUD operations on users

UserManager<IdentityUser>

```
<ItemGroup>
  ...

  <PackageReference
    Include="Microsoft.AspNetCore.Identity.EntityFrameworkCore"
    Version="1.1.1" />
  ...
</ItemGroup>
```

Adding the Correct Packages

Enabling Identity

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddDbContext<AppDbContext>(
      options =>
        options.UseSqlServer
         (_configurationRoot.GetConnectionString("DefaultConnection")));


    services.AddIdentity<IdentityUser, IdentityRole>()
      .AddEntityFrameworkStores<AppDbContext>();
}
```

```
public void Configure(IApplicationBuilder app)
{
    app.UseIdentity();
}
```

Enabling Identity

Enumerating Users Using UserManager

```
private readonly UserManager<IdentityUser> _userManager;

public IActionResult UserManagement()
{
    var users = _userManager.Users;

    return View(users);

}
```

```
IdentityResult result =
  await _userManager.CreateAsync(user, addUserViewModel.Password);
```

Adding a User

Applying Validation on the Created Users

```
services.AddIdentity<IdentityUser, IdentityRole>(options =>
{
    options.Password.RequiredLength = 8;

    options.Password.RequireNonAlphanumeric = true;

    options.Password.RequireUppercase = true;

    options.User.RequireUniqueEmail = true;

}).AddEntityFrameworkStores<AppDbContext>();
```

Demo

Creating a user overview

Adding user management

Extending the IdentityUser

IdentityUser

```
...public class IdentityUser<TKey, TUserClaim, TUserRole, TUserLogin> where TKey : IEquatable<TKey>
{
    ...public IdentityUser();
    ...public IdentityUser(string userName);

    ...public virtual int AccessFailedCount { get; set; }
    ...public virtual ICollection<TUserClaim> Claims { get; }
    ...public virtual string ConcurrencyStamp { get; set; }
    ...public virtual string Email { get; set; }
    ...public virtual bool EmailConfirmed { get; set; }
    ...public virtual TKey Id { get; set; }
    ...public virtual bool LockoutEnabled { get; set; }
    ...public virtual DateTimeOffset? LockoutEnd { get; set; }
    ...public virtual ICollection<TUserLogin> Logins { get; }
    ...public virtual string NormalizedEmail { get; set; }
    ...public virtual string NormalizedUserName { get; set; }
    ...public virtual string PasswordHash { get; set; }
    ...public virtual string PhoneNumber { get; set; }
    ...public virtual bool PhoneNumberConfirmed { get; set; }
    ...public virtual ICollection<TUserRole> Roles { get; }
    ...public virtual string SecurityStamp { get; set; }
    ...public virtual bool TwoFactorEnabled { get; set; }
    ...public virtual string UserName { get; set; }

    ...public override string ToString();
}
```

```
public class ApplicationUser: IdentityUser
{
    public DateTime Birthdate { get; set; }
    public string City { get; set; }
    public string Country { get; set; }
}
```

Adding Extra Properties

```
services.AddIdentity<ApplicationUser, IdentityRole>()
  .AddEntityFrameworkStores<AppDbContext>();
```

Custom User Replaces IdentityUser

Demo

Extending the IdentityUser class to capture more properties

A question from Bethany

- I need to manage all these users in some sort of groups since they will typically have the same tasks within my site

Role Management

RoleManager

Creation of roles

Adding users to roles

[Authorize]

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

Role-based Authorization

```
[Authorize(Roles = "Administrators")]
[Authorize(Roles = "SomeOtherRole")]
public class AdminController : Controller

{

}
```

Role-based Authorization

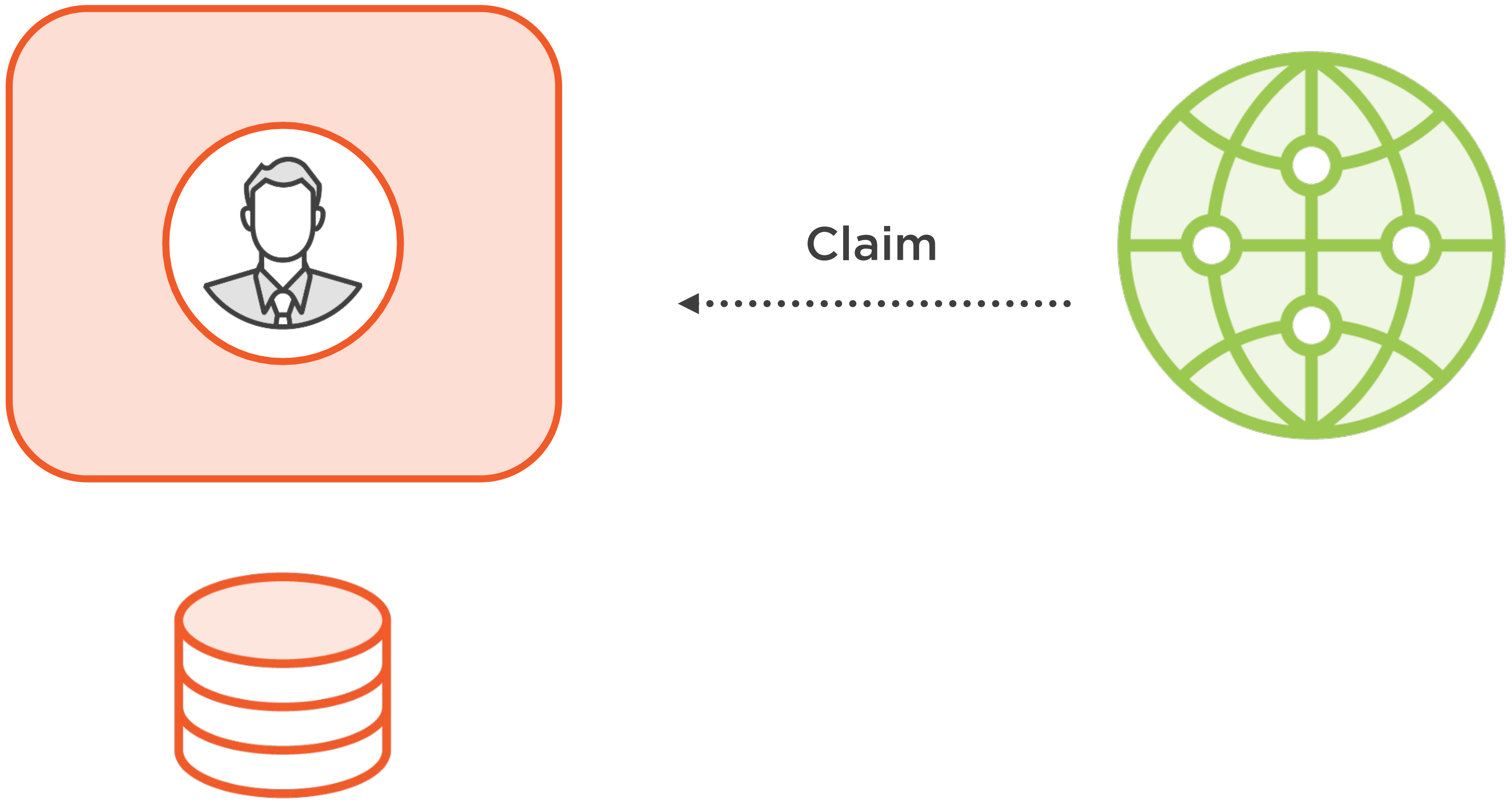
Demo

Adding role management

Adding users to roles

Role-based authorization

Claims-based Authorization



Claim

at

o

external

Policy-based

```
services.AddAuthorization(options =>
{

   options.AddPolicy("DeletePie", policy =>
     policy.RequireClaim("Delete Pie"));

});
```

Registering the Policy

```
[Authorize(Policy = "DeletePie")]
public class AdminController : Controller
{

    ...
}
```

Applying the Policy on a Controller or Action

```
[Authorize(Roles = "Administrators")]
[Authorize(Policy = "DeletePie")]
[Authorize(Policy = "AddPie")]
public class AdminController : Controller
{

    ...

}
```

Combining Several Policies

Demo

Adding claims-based authorization

Built-in, pre-configured policy

- RequireClaim

Custom policy

- Authorization requirement(s)

The Requirement

```
public class MinimumOrderAgeRequirement:
IAuthorizationRequirement
{
    private readonly int _minimumOrderAge;

    public MinimumOrderAgeRequirement(int minimumOrderAge)
    {
        _minimumOrderAge = minimumOrderAge;

    }

}
```

```
public class MinimumOrderAgeRequirement:
  AuthorizationHandler<MinimumOrderAgeRequirement>

{

    ...
}
```

The Authorization Handler

Evaluates properties of the requirement(s)

```
services.AddAuthorization(options =>
{
   options.AddPolicy("MinimumOrderAge", policy =>
     policy.Requirements.Add(
       new MinimumOrderAgeRequirement(18)));

});
```

Using the Custom Requirement

Demo

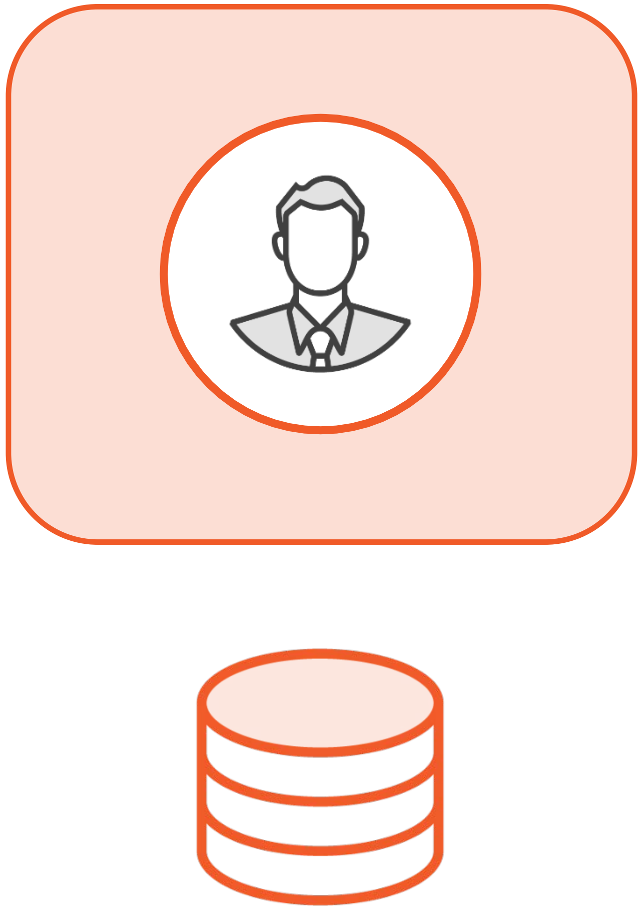
Using a custom policy

A question from Bethany

- I want people to use an existing Google or Facebook account to log in to my application

Adding Third-party Authentication

Third-party Authentication



Claim

Google

Facebook

Microsoft

Twitter

Advantages

eg

cr

tication

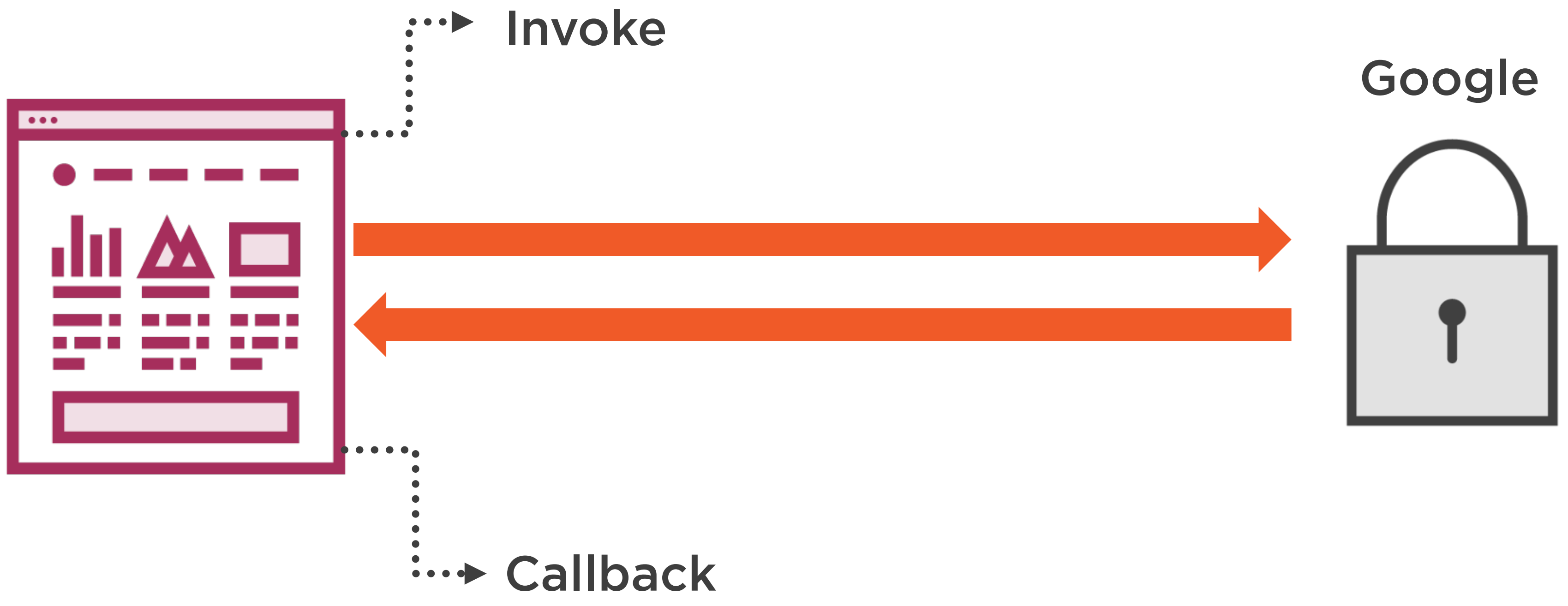
Support built-in

```xml
<ItemGroup>

   <PackageReference
      Include="Microsoft.AspNetCore.Authentication.Google"
      Version="1.1.1" />

</ItemGroup>
```

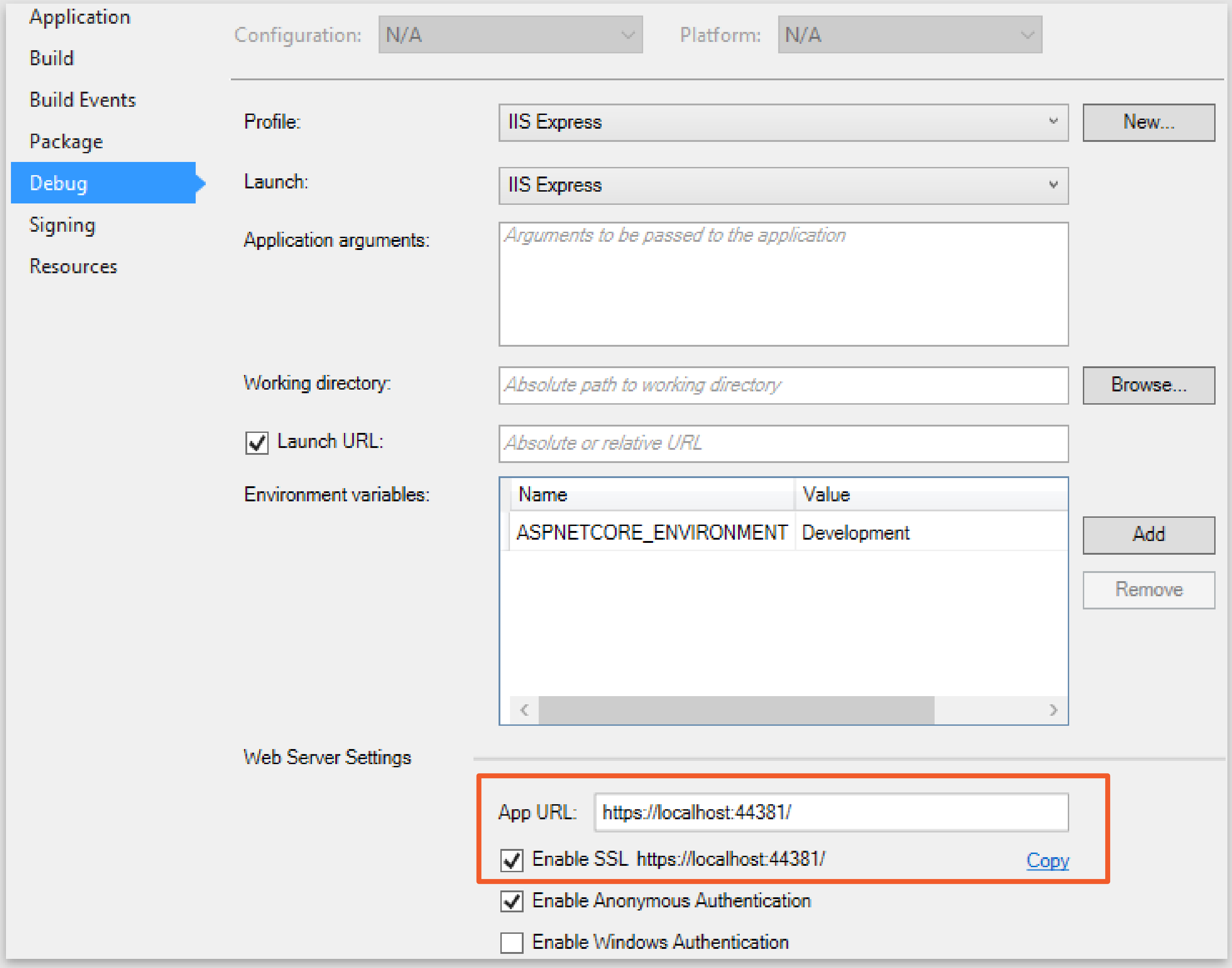
It Starts with a Package

```
app.UseGoogleAuthentication(new GoogleOptions
{
    ClientId = "...",

    ClientSecret = "..."

});
```

Using Google Authentication



SSL Is Required



Configuration: N/A
Platform: N/A

Profile: IIS Express
New...

Launch: IIS Express

Application arguments: Arguments to be passed to the application

Working directory: Absolute path to working directory
Browse...

☑ Launch URL: Absolute or relative URL

Environment variables:

Name	Value
ASPNETCORE_ENVIRONMENT	Development

Web Server Settings

App URL: https://localhost:44381/

☑ Enable SSL https://localhost:44381/ Copy

☑ Enable Anonymous Authentication

☐ Enable Windows Authentication

Demo

Enabling third-party authentication

Summary

ASP.NET Identity offers a lot of functionality

Built-in support for

- Role
- Claims
- Third-party authentication

Up next:

Securing our site against attacks