20251020Speeding Up Local...

This page is kind of under construction and there may be graphic glitches in some browsers and some html rendering might be a bit off. It'll get better.

Speeding Up Local Development with a Fake OpenID Authentication Handler

Categories Development Tags .net OpenID CMS Optimizely

Setting up OpenID authentication can be painful in development environments – especially when the identity server isn't publicly accessible or you're still waiting on client configuration.

To solve this, we created a **local authentication scheme** that behaves like OpenID but doesn't require any network connection or external setup. It lets developers instantly sign in as a fake user with predefined roles – perfect for local testing and remote development.

Here's the simplified implementation:

```
// usings
using Microsoft.AspNetCore.Authentication;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Options;
using System.Security.Claims;
using System.Text.Encodings.Web;
using System.Threading.Tasks;

internal class LocalAuthenticationSchemeOptions : AuthenticationSchemeOptions { }

internal class LocalAuthenticationSchemeHandler(
    IOptionsMonitor options,
    ILoggerFactory logger,
    UrlEncoder encoder)
    : AuthenticationHandler(options, logger, encoder)
{
    protected override async Task HandleAuthenticateAsync()
    {
        var claims = new[]
        {
            // add any claim you need, these are for Optimizely CMS
            new Claim(ClaimTypes.Role, "Admins"),
            new Claim(ClaimTypes.Role, "CmsAdmins"),
            new Claim(ClaimTypes.Role, "CmsEditors"),
            new Claim(ClaimTypes.Name, "John Doe")
        };

        var principal = new ClaimsPrincipal(new ClaimsIdentity(claims, "Local"));
        var ticket = new AuthenticationTicket(principal, Scheme.Name);

        return await Task.FromResult(AuthenticateResult.Success(ticket));
    }
}
```

For Optimizely you'll need to ensure that the claims added correspond with what configs you have in you appsettings, see https://docs.developers.optimizely.com/content-management-system/docs/virtual-roles.

And we wire it up conditionally during startup:

```
if (environment.IsDevelopment())
{
    services.AddAuthentication("LocalAuthentication")
        .AddScheme<LocalAuthenticationSchemeOptions, LocalAuthenticationSchemeHandler>(
            "LocalAuthentication",
            opts => { });
}
else
{
    // Your OpenID-implementation, may be services.AddAuthentication...
    services.AddOpenIdServicesConfiguration(configuration, logger);
}
```

This setup means:

- **Local devs can log in instantly**, without OpenID.
- **Full control over user identity and roles** — test as any role.
- **No dependency on external servers**, making remote/offline dev possible.
- **Production remains secure**, since the handler is only active in `Development` mode.

It's a small addition, but it dramatically simplifies authentication workflows during development. Instead of waiting for OpenID configuration or tunneling into internal networks, developers can jump straight into building features.