B8

My Curiosity In Business Context At Labman

Arthur Clarkson

29th October 2024

For user identification, our system adds `userid`, `username`, and `displayname` as claims in a cookie, which is included in every HTTP request as a header. The server then reads this cookie to identify the user making the request. Our standard methods for retrieving these details look like this:

```
protected string CurrentUsername()
{
        return _httpContextAccessor?.HttpContext?.User.CurrentUsername();
}

protected string CurrentDisplayname()
{
        return _httpContextAccessor?.HttpContext?.User.CurrentDisplayName();
}

protected int CurrentUserId()
{
        var username = CurrentUsername();
        if (string.IsNullOrEmpty(username)) { return 0; }

        return _db.Users.FirstOrDefault(u => u.Username == username)?.Id ?? 0;
}

protected User CurrentUser()
{
        var username = CurrentUsername();
        return _db.Users.FirstOrDefault(u => u.Username == username);
}
```

These methods have been in place for over five years, but I noticed performance issues with `CurrentUserId()`, which retrieves the `username` claim and then queries the database to return the corresponding `userid`. This approach, involving a database call for each request, is inefficient—especially given that we can obtain the `userid` directly from the claims.

Here's a more optimized approach that reduces bandwidth and computation time significantly:

```
protected int CurrentUserId()
{
        string userIdClaim = _httpContextAccessor?.HttpContext?.User.CurrentUserId();
        return int.TryParse(userIdClaim, out var userId) ? userId : 0;
}
```

This alternative approach leverages the existing claim data, avoiding unnecessary database calls and parsing the `userid` directly from the claims. With this enhancement, which eliminates an over-reliance on database resources, we save gigabytes of bandwidth and reduce hours of computational overhead daily.

This illustrates a response driven by curiosity, where I investigated the system’s performance bottlenecks and identified an area for improvement. By implementing a direct claim parsing approach, I leveraged both business context and technical insight to achieve a more efficient solution.