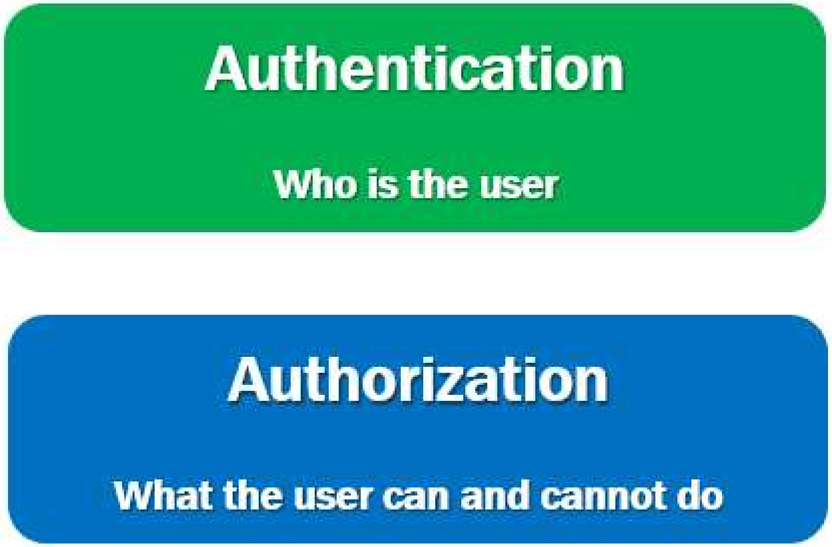
are-restful-apis)
Create REST API
(https://www.pragimtech.com/blog
asp.net-core-rest-api)
REST API DbContext
(https://www.pragimtech.com/blog
core-rest-api-dbcontext)
Repository Pattern in REST API
(https://www.pragimtech.com/blog
api-repository-pattern)
REST API Get List
(https://www.pragimtech.com/blog
core-rest-api-list-of-resources)
REST API Get By Id
(https://www.pragimtech.com/blog
core-rest-api-get-by-id)
Post in REST API
(https://www.pragimtech.com/blog
in-aspnet-core-rest-api)
REST API Model Validation
(https://www.pragimtech.com/blog
api-model-validation)
Put in REST API
(https://www.pragimtech.com/blog
in-asp.net-core-rest-api)
Delete in REST API
(https://www.pragimtech.com/blog
in-asp.net-core-rest-api)
Search in REST API
(https://www.pragimtech.com/blog

Authorization in Blazor

In this video we will discuss Authorization in Blazor, specifically how to show and hide UI elements depending on the authentication state of the user.

Authorization in ASP.NET Core



Authentication

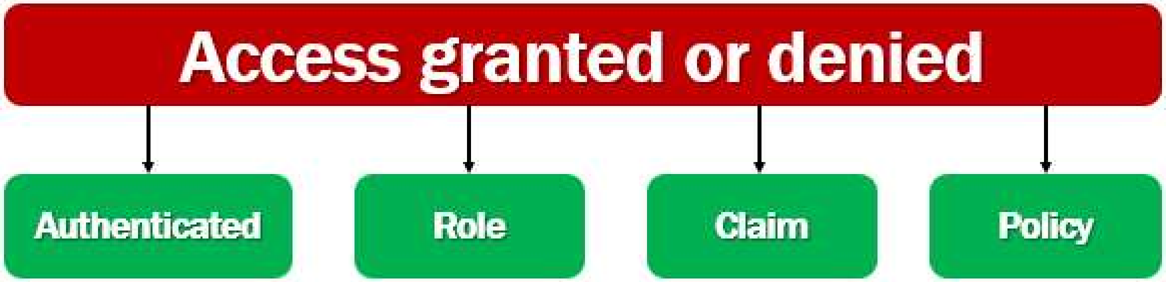
Who is the user

Authorization

What the user can and cannot do

- Authentication is the process of identifying who the user is.
- Authorization is the process of identifying what the user can and cannot do.

- For example, if the logged in user, is an administrator, he may be able to Create Read, Update and Delete orders, where as a normal user may only view orders but not Create, Update or Delete orders.



Access is typically granted or denied based on whether:

- A user is authenticated (logged in).
- A user is in a role.
- A user has a claim.
- A policy is satisfied.

We discussed all these concepts i.e authorization using roles, claims and policies in detail in ASP.NET Core tutorial for beginners course (https://www.pragimtech.com/courses/asp-net-core-mvc-tutorial-for-beginners/). W use roles, claims and policies the same way in ASP.NET Core MVC apps, razor pages and Blazor. If you are new to these concepts please check out videos starting at Part 77 (https://youtu.be/NV734cJdZts) from ASP.NET Core tutorial for beginners (https://www.youtube.com/playlist?list=PL6n9fhu94yhVkdrusLaQsfERmL_Jh4XmU) course.

Blazor AuthorizeView component

- In Blazor we use `AuthorizeView` component to show or hide UI elements depending on whether the user is authorized to see it.
- In this example, `AuthorizeView` component is used in it's simplest form, withou any parameters (i.e roles or policies), so, it only checks if the user is

authenticated.

- If the user is authenticated, then the content in `<Authorized>` component is displayed, otherwise, the content in `<NotAuthorized>` component is displayed.

```
<AuthorizeView>
    <Authorized>
        This content is displayed only if the user is Authorized
    </Authorized>
    <NotAuthorized>
        This content is displayed if the user is Not Authorized
    </NotAuthorized>
</AuthorizeView>
```

CascadingAuthenticationState component

If you get the following error

InvalidOperationException: Authorization requires a cascading parameter of type Task<AuthenticationState>. Consider using CascadingAuthenticationState to suppl this.

The error message is pretty good. It clearly tells us what needs to be done. Authentication state must be provided to the application and we do this by wrapping by the `<Router>` component with `<CascadingAuthenticationState>` component. The `<Router>` component is in `App.razor` file.

```
<CascadingAuthenticationState>
    <Router AppAssembly="@typeof(Program).Assembly">
        <Found Context="routeData">
            <RouteView RouteData="@routeData" DefaultLayout="@typeof(MainLayout)" /
        </Found>
        <NotFound>
            <LayoutView Layout="@typeof(MainLayout)">
                <p>Sorry, there's nothing at this address.</p>
            </LayoutView>
        </NotFound>
    </Router>
</CascadingAuthenticationState>
```

in-asp.net-core-rest-api)
Call REST API from Blazor
are-restful-apis)
Create REST API
(https://www.pragimtech.com/blo
asp.net-core-rest-api)
REST API DbContext
(https://www.pragimtech.com/blo
core-rest-api-dbcontext)
Repository Pattern in REST API
(https://www.pragimtech.com/blo
api-repository-pattern)
REST API Get List
(https://www.pragimtech.com/blo
core-rest-api-list-of-resources)
REST API Get By Id
(https://www.pragimtech.com/blo
core-rest-api-get-by-id)
Post in REST API
(https://www.pragimtech.com/blo
in-aspnet-core-rest-api)
REST API Model Validation
(https://www.pragimtech.com/blo
api-model-validation)
Put in REST API
(https://www.pragimtech.com/blo
in-asp.net-core-rest-api)
Delete in REST API
(https://www.pragimtech.com/blo
in-asp.net-core-rest-api)
Search in REST API
(https://www.pragimtech.com/blo

Role or policy based authorization in Blazor

Both role-based and policy-based authorization is supported in Blazor. For role-based authorization, use the `Roles` parameter.

```
<AuthorizeView Roles="administrator, manager">
    <p>Displayed if the logged in user is in administrator or manager role</p>
</AuthorizeView>
```

For policy-based authorization, use the `Policy` parameter:.

```
<AuthorizeView Policy="admin-policy">
    <p>Displayed if the logged in user staisfies admin-policy</p>
</AuthorizeView>
```

AuthorizeView component context variable

- `AuthorizeView` component exposes a `context` variable.
- The variable type is `AuthenticationState`.
- Use it to access autneticated user information.

In the following example, we are using the `context` variable to displayed the logged-in username.

in-asp.net-core-rest-api)

Call REST API from Blazor

are-restful-apis)

Create REST API
(https://www.pragimtech.com/blo
asp.net-core-rest-api)

REST API DbContext
(https://www.pragimtech.com/blo
core-rest-api-dbcontext)

Repository Pattern in REST API
(https://www.pragimtech.com/blo
api-repository-pattern)

REST API Get List
(https://www.pragimtech.com/blo
core-rest-api-list-of-resources)

REST API Get By Id
(https://www.pragimtech.com/blo
core-rest-api-get-by-id)

Post in REST API
(https://www.pragimtech.com/blo
in-aspnet-core-rest-api)

REST API Model Validation
(https://www.pragimtech.com/blo
api-model-validation)

Put in REST API
(https://www.pragimtech.com/blo
in-asp.net-core-rest-api)

Delete in REST API
(https://www.pragimtech.com/blo
in-asp.net-core-rest-api)

Search in REST API
(https://www.pragimtech.com/blo

```
<AuthorizeView>
    <Authorized>
        <li class="nav-item px-3">
            <NavLink class="nav-link" href="/identity/account/logout" Match="NavLin
                <span class="oi oi-power-standby" aria-hidden="true"></span>
                Logout (@context.User.Identity.)
            </NavLink>
        </li>
    </Authorized>
</AuthorizeView>
```

in-asp.net-core-rest-api)

Call REST API from Blazor

are-restful-apis)

Create REST API
(https://www.pragimtech.com/blo
asp.net-core-rest-api)

REST API DbContext
(https://www.pragimtech.com/blo
core-rest-api-dbcontext)

Repository Pattern in REST API
(https://www.pragimtech.com/blo
api-repository-pattern)

REST API Get List
(https://www.pragimtech.com/blo
core-rest-api-list-of-resources)

REST API Get By Id
(https://www.pragimtech.com/blo
core-rest-api-get-by-id)

Post in REST API
(https://www.pragimtech.com/blo
in-aspnet-core-rest-api)

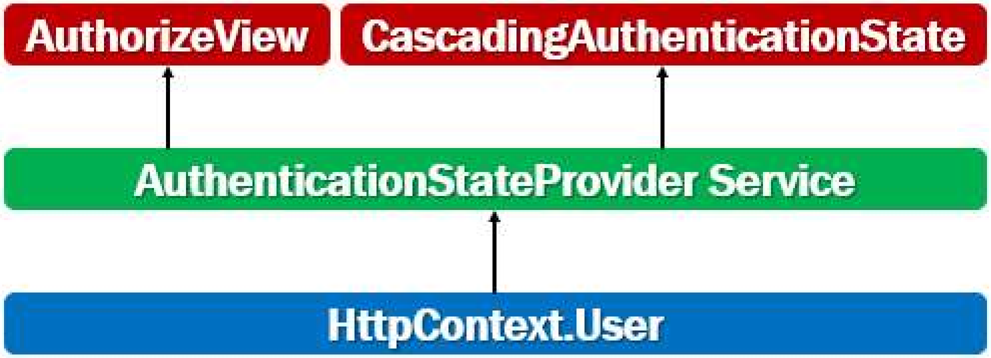
REST API Model Validation
(https://www.pragimtech.com/blo
api-model-validation)

Put in REST API
(https://www.pragimtech.com/blo
in-asp.net-core-rest-api)

Delete in REST API
(https://www.pragimtech.com/blo
in-asp.net-core-rest-api)

Search in REST API
(https://www.pragimtech.com/blo

AuthenticationStateProvider service


AuthorizeView
CascadingAuthenticationState
AuthenticationStateProvider Service
HttpContext.User

- Blazor has a built-in service called `AuthenticationStateProvider` service.
- This service obtains authentication state data from ASP.NET Core's `HttpContext.User`.
- This is how authentication state integrates with existing ASP.NET Core authentication mechanisms.
- It is this service that is used by `AuthorizeView` component and `CascadingAuthenticationState` component to get the authentication state.
- Don't use `AuthenticationStateProvider` directly. Use the `AuthorizeView` component.
- The main drawback to using `AuthenticationStateProvider` directly is that the component isn't notified automatically if the underlying authentication state dat changes.

Prev (https://www.pragimtech.com/blog/blazor/blazor-logout-page)
(https://www.pragimtech.com/blog/blazor/blazor-authorize-attribute) Next

in-asp.net-core-rest-api)

Call REST API from Blazor

are-restful-apis)

Create REST API
(https://www.pragimtech.com/blo
asp.net-core-rest-api)

REST API DbContext
(https://www.pragimtech.com/blo
core-rest-api-dbcontext)

Repository Pattern in REST API
(https://www.pragimtech.com/blo
api-repository-pattern)

REST API Get List
(https://www.pragimtech.com/blo
core-rest-api-list-of-resources)

REST API Get By Id
(https://www.pragimtech.com/blo
core-rest-api-get-by-id)

Post in REST API
(https://www.pragimtech.com/blo
in-aspnet-core-rest-api)

REST API Model Validation
(https://www.pragimtech.com/blo
api-model-validation)

Put in REST API
(https://www.pragimtech.com/blo
in-asp.net-core-rest-api)

Delete in REST API
(https://www.pragimtech.com/blo
in-asp.net-core-rest-api)

Search in REST API
(https://www.pragimtech.com/blo