Using Claims-based Authorization

Paul D. Sheriff

BUSINESS SOLUTIONS ARCHITECT, FAIRWAY TECHNOLOGIES, INC.

www.fairwaytech.com psheriff@fairwaytech.com

Goals

Use array of claims instead of properties

Modify Angular classes

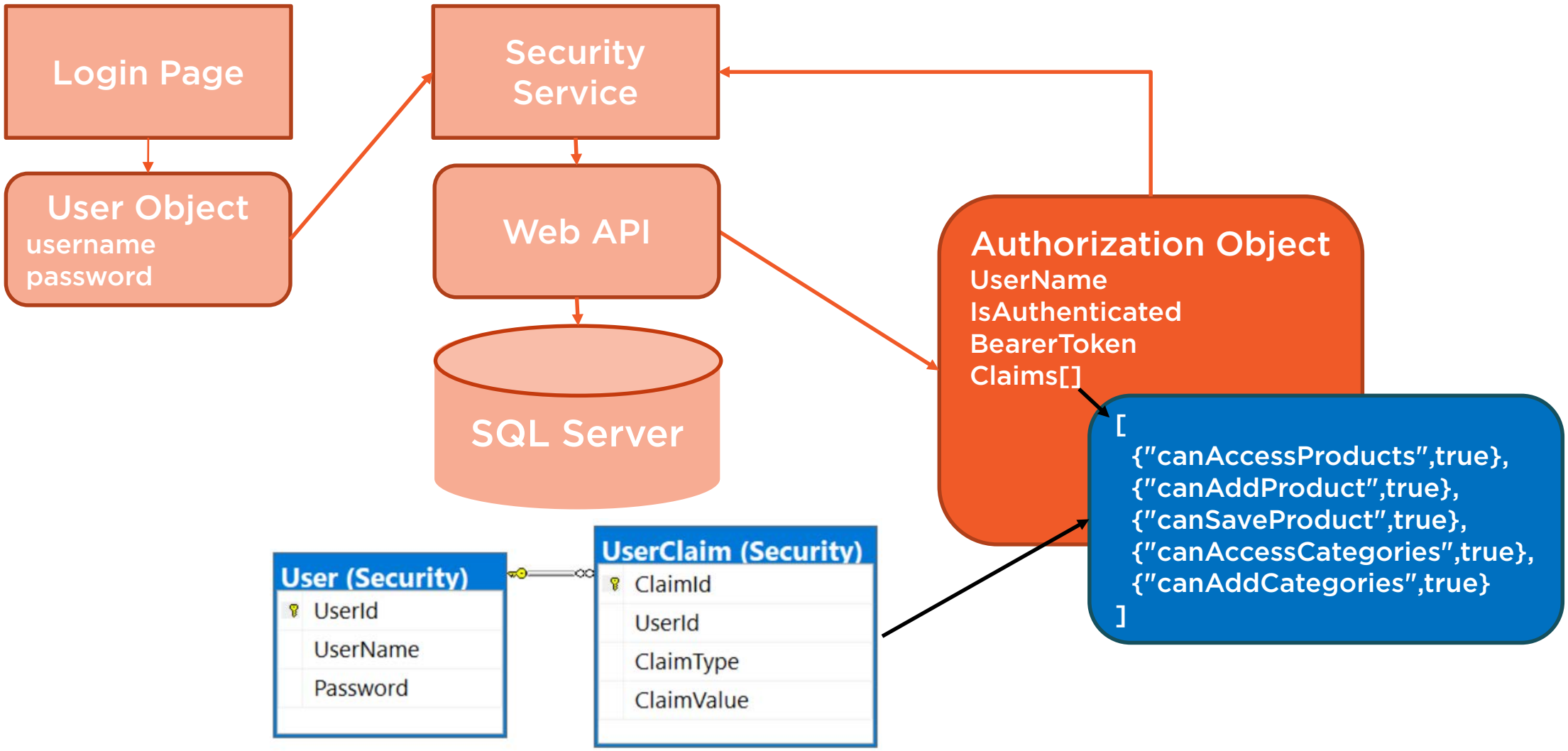
Modify C# classes

Create structural directive

- *hasClaim="'canAccessProducts'"

Use Array of Claims

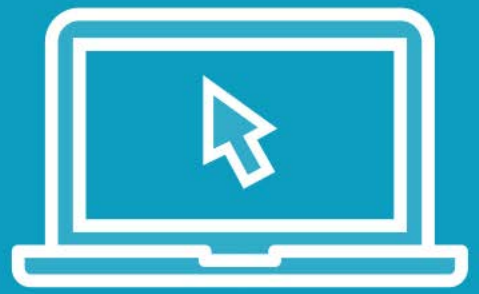
Security Architecture Using Claims



Use Array of Claims Instead of Properties

- Add AppUserClaim class
- Add claims array to AppUserAuth
- Modify resetSecurityObject method to clear claims array
- Add isClaimValid() method
- Add hasClaim() method
- Modify route guard

Demo



Modify Angular classes

Modify Server-side Classes

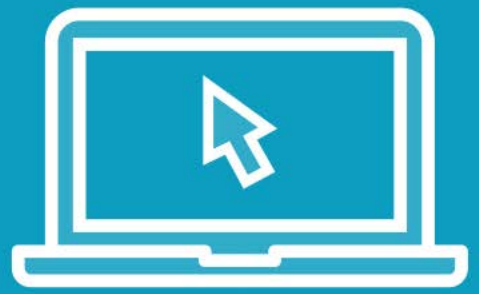
Modify Server-side Classes

Add list of claims to AppUserAuth

Assign list of claims in Build UserAuthObject() method

Add claims to jwtClaims array in BuildJwtToken() method

Demo



Modify server-side classes

View the user auth object returned

Create Structural Directive

Create structural directive named "hasClaim"

Removes element if claim is not valid

```
<button class="btn btn-primary" (click)="addProduct()"
        *hasClaim="'CanAddProduct'">
  Add New Product
</button>
```

Create a directive

Add selector

Add @Input property

Check for claim

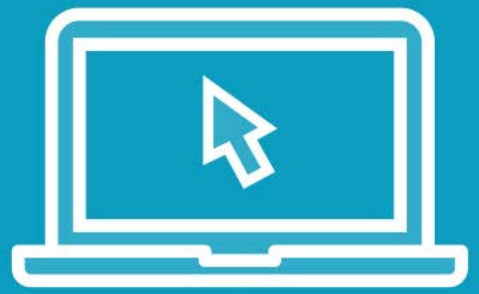
Add element to DOM

Remove element from DOM

```
@Directive({
  selector: '[hasClaim]'
})
export class HasClaimDirective {
  constructor(
    private templateRef: TemplateRef<any>,
    private viewContainer: ViewContainerRef,
    private securityService: SecurityService) { }

  @Input() set hasClaim(claimType: any) {
    if (this.securityService.hasClaim(claimType)) {
      // Add template to DOM
      this.viewContainer.createEmbeddedView(this.templateRef);
    } else {
      // Remove template from DOM
      this.viewContainer.clear();
    }
  }
}
```

Demo



Create structural directive

Use on button elements

Secure Menus

Not allowed to have two structural directives on one element

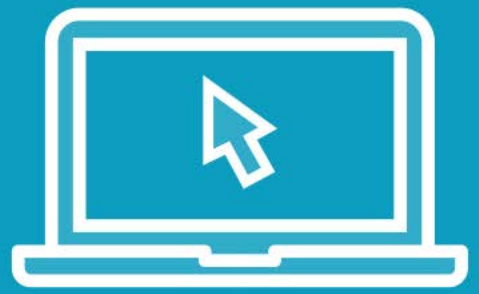
Wrap menu in ng-container

isAuthenticated

Use *hasClaim on menu

```
<li>
  <ng-container *ngIf="securityObject.isAuthenticated">
    <a routerLink="/products" *hasClaim="'canAccessProducts'">
      Products
    </a>
  </ng-container>
</li>
```

Demo



Secure menu items

Multiple Claims

Allow more than one claim per UI element

Pass an array of claims

```
<button class="btn btn-primary" (click)="addProduct()"
        *hasClaim="['CanAddProduct', 'CanAccessCategories']">
  Add New Product
</button>
```

Modify the hasClaim() method

Check *claimType* parameter

If array, convert to array

Loop through

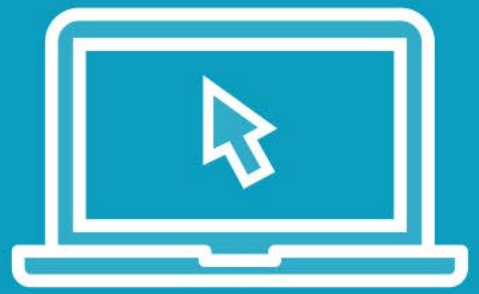
If one is successful, return true

```
hasClaim(claimType: any, claimValue?: any) {
  let ret: boolean = false;

  // See if an array of values was passed in.
  if (typeof claimType === "string") {
    ret = this.isClaimValid(claimType, claimValue);
  }
  else {
    let claims: string[] = claimType;
    if (claims) {
      for (let index = 0; index < claims.length; index++) {
        ret = this.isClaimValid(claims[index]);
        // If one is successful, then let them in
        if (ret) {
          break;
        }
      }
    }
  }

  return ret;
}
```

Demo



Handle multiple claims

Secure other buttons

Remove code from components

Summary

- Added array of claims to auth object
- Returned array of claims from Web API
- Used structural directive to check claims
- Support for multiple claims
- Simplified code
- You can add a role array and implement role-based just like claims-based

Course Summary

- Authenticated users
- Secured UI elements and routes
- Used JSON Web Tokens to secure Web API
- Learned to use bearer tokens
- Used claims-based authorization
- Created structural directive to simplify code

I hope you enjoyed this course!

Paul D. Sheriff

Business Solutions Architect, Fairway Technologies, Inc.

www.fairwaytech.com **psheriff@fairwaytech.com**