Secure UI Elements and Guard Routes

Paul D. Sheriff

BUSINESS SOLUTIONS ARCHITECT, FAIRWAY TECHNOLOGIES, INC.

www.fairwaytech.com psheriff@fairwaytech.com

Goals

- Make menus visible/invisible
- Make buttons visible/invisible
- Prevent direct navigation
- Redirect to login page

Secure Menus

Modify app.component.ts

Add AppUserAuth object

Inject security service

Retrieve security object

Add logout() method

```
@Component({
  selector: 'ptc-root',
  templateUrl: './app.component.html'
})
export class AppComponent {
  title: string = "Paul's Training Company";
  securityObject: AppUserAuth = null;

  constructor(private securityService: SecurityService) {
    this.securityObject = securityService.securityObject;
  }

  logout(): void {
    this.securityService.logout();
  }
}
```

Modify app.component.html

Add structural directives

Bind to properties in AppUserAuth object

```
<li>
  <a routerLink="/products"
     *ngIf="securityObject.canAccessProducts">
    Products
  </a>
</li>
<li>
  <a routerLink="/categories"
     *ngIf="securityObject.canAccessCategories">
    Categories
  </a>
</li>
```

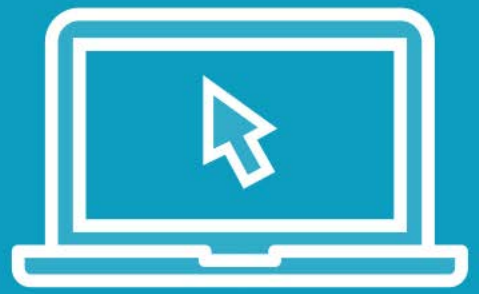
Toggle Login/Logout menus

If not authenticated, display Login

If authenticated, display Logout with user name

```
<ul class="nav navbar-nav navbar-right">
  <li>
    <a routerLink="login"
      *ngIf="!securityObject.isAuthenticated">
      Login
    </a>
    <a href="#" (onclick)="logout()"
      *ngIf="securityObject.isAuthenticated">
      Logout {{securityObject.userName}}
    </a>
  </li>
</ul>
```

Demo



Secure menus

Secure Buttons

Modify product-list.component.ts

Add AppUserAuth object

Inject security service

Retrieve security object

```
@Component({
  templateUrl: './product-list.component.html'
})
export class ProductListComponent implements OnInit {
  products: Product[];
  securityObject: AppUserAuth = null;

  constructor(private productService: ProductService,
    private router: Router,
    private securityService: SecurityService) {
    this.securityObject = securityService.securityObject;
    }
```

Modify product-list.component.html

Add structural directive to Add New Product button

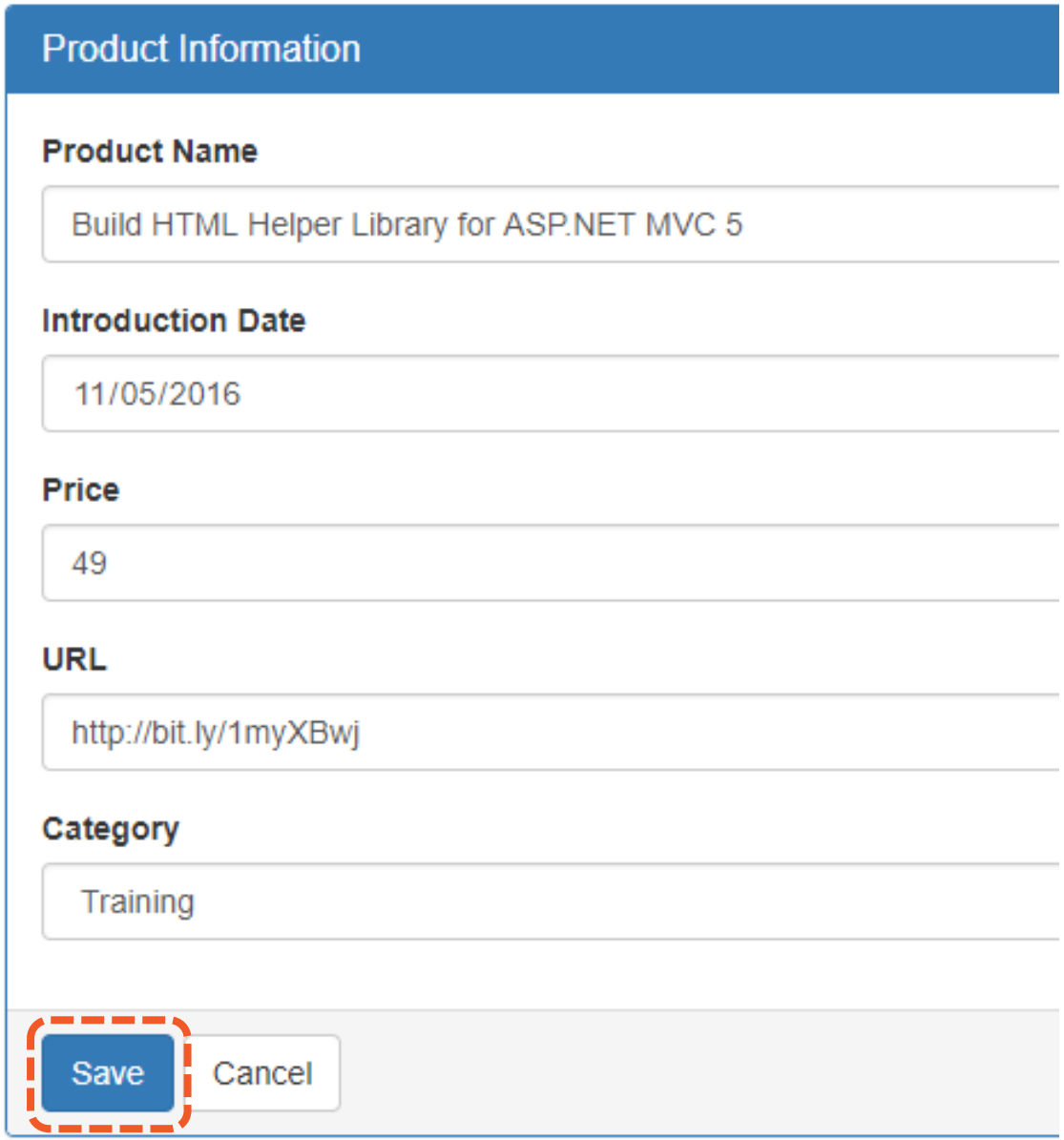
```
<button class="btn btn-primary"
        (click)="addProduct()"
        *ngIf="securityObject.canAddProduct">
  Add New Product
</button>
```

Secure the Save button on the product detail page

Add security object property

Inject security service

Retrieve security object

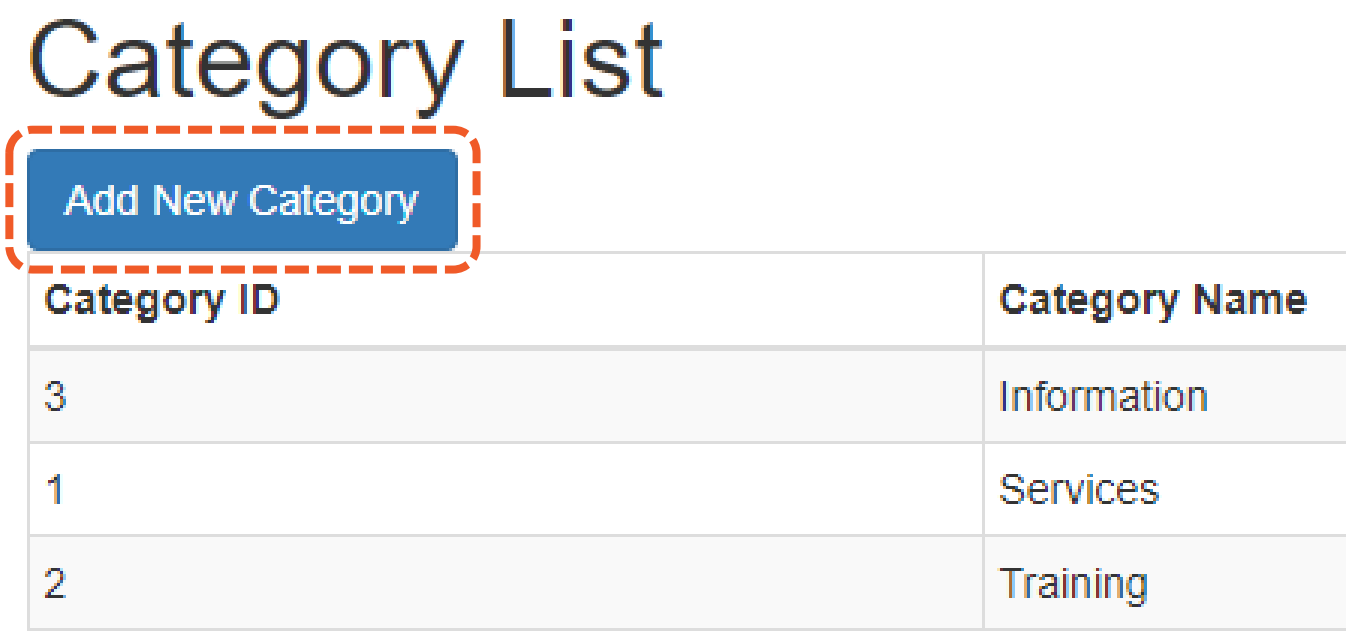


Secure the Add New Category button on the category list page

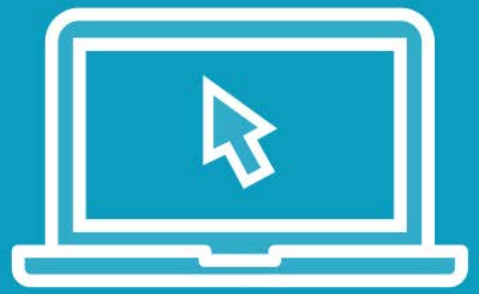
Add security object property

Inject security service

Retrieve security object



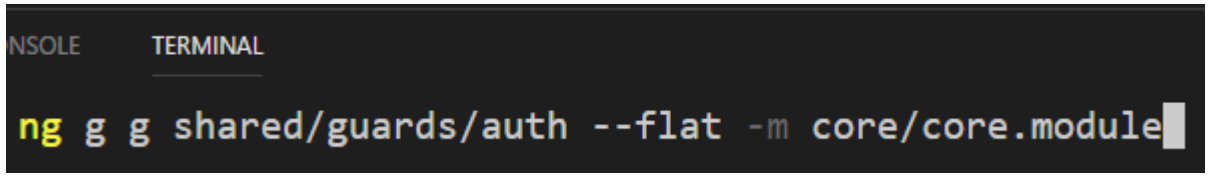
Demo



Secure buttons

Route Guards

Create auth guard



Open app-routing.module.ts

Add guard to routes to secure

Pass in data object

Set *claimType* to property name to check

```
{
  path: 'products',
  component: ProductListComponent,
  canActivate: [AuthGuard],
  data: { claimType: 'canAccessProducts' }
},
{
  path: 'productDetail/:id',
  component: ProductDetailComponent,
  canActivate: [AuthGuard],
  data: { claimType: 'canAccessProducts' }
},
{
  path: 'categories',
  component: CategoryListComponent,
  canActivate: [AuthGuard],
  data: { claimType: 'canAccessCategories' }
},
```

Inject security service

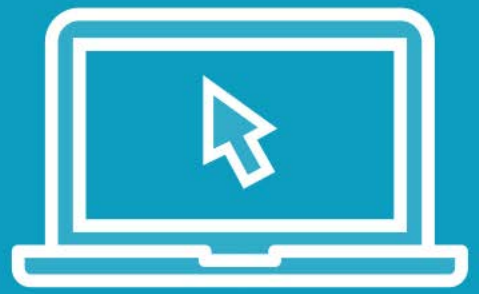
Retrieve claim type from *data* property

Check *isAuthenticated*

Check if claim property is true

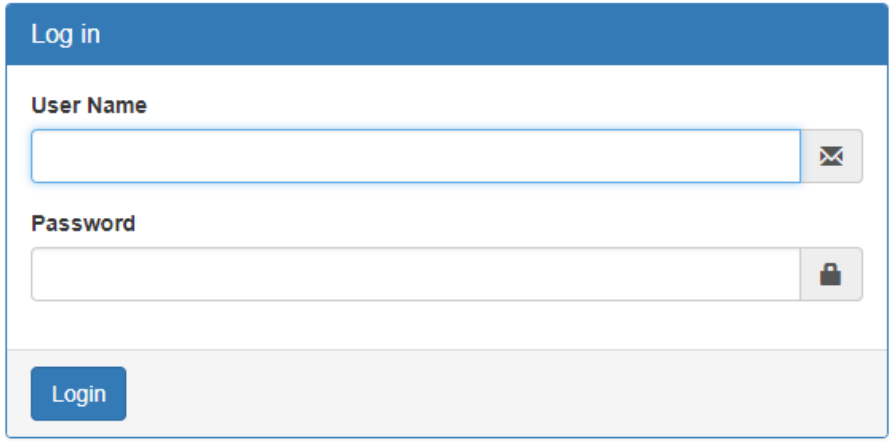
```
@Injectable()
export class AuthGuard implements CanActivate {
  constructor(private securityService: SecurityService) { }

  canActivate(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean> | Promise<
    // Get property name on security object to check
    let claimType: string = next.data["claimType"];

    return this.securityService.securityObject.isAuthenticated
      && this.securityService.securityObject[claimType];
  }
}
```

Demo



Route guards

Redirect to Login Page



If not authorized for route, redirect to login page

- Pass page requested to login page
- Store requested page in login component

Ask user to enter credentials

- If valid credentials redirect back to requested page
- If not valid, redirect back to login page

Modify auth-guard.ts

Inject Router

Modify return to an if statement

If authenticated return true

Otherwise grab state.url for *queryParams* object

Navigate to login page

```
@Injectable()
export class AuthGuard implements CanActivate {
  constructor(private securityService: SecurityService,
    private router: Router) { }

  canActivate(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean> | Promi
    // Get property name on security object to check
    let claimType: string = next.data["claimType"];

    if (this.securityService.securityObject.isAuthenticated
      && this.securityService.securityObject[claimType]) {
      return true;
    }
    else {
      this.router.navigate(['login'],
        { queryParams: { returnUrl: state.url } });
      return false;
    }
  }
}
```

Modify login.component.ts

Add *returnUrl* property

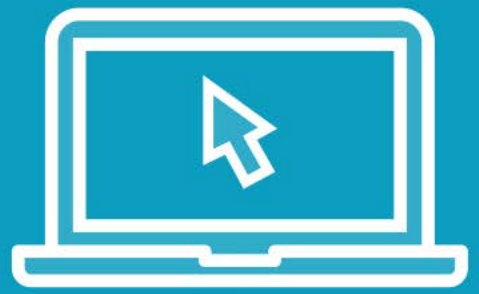
Inject ActivatedRoute and Router

Set *returnUrl*

If success, navigate to *returnUrl*

```
export class LoginComponent implements OnInit {
  user: AppUser = new AppUser();
  securityObject: AppUserAuth = null;
  returnUrl: string;

  constructor(private securityService: SecurityService,
    private route: ActivatedRoute,
    private router: Router) { }

  ngOnInit() {
    this.returnUrl =
      this.route.snapshot.queryParamMap.get('returnUrl');
  }

  login() {
    this.securityService.login(this.user)
      .subscribe(resp => {
      this.securityObject = resp;
        if (this.returnUrl) {
          this.router.navigateByUrl(this.returnUrl);
        }
      });
  }
}
```

Demo



Redirect to login page

Summary

- Added structural directives to secure menus and buttons
- Created route guard to protect navigation
- Redirected to login page if not authorized
- Redirected back to requested page

Coming up in the next module...

User authentication via Web API

Return user authorization object