Call Web API to Authenticate and Authorize

Paul D. Sheriff

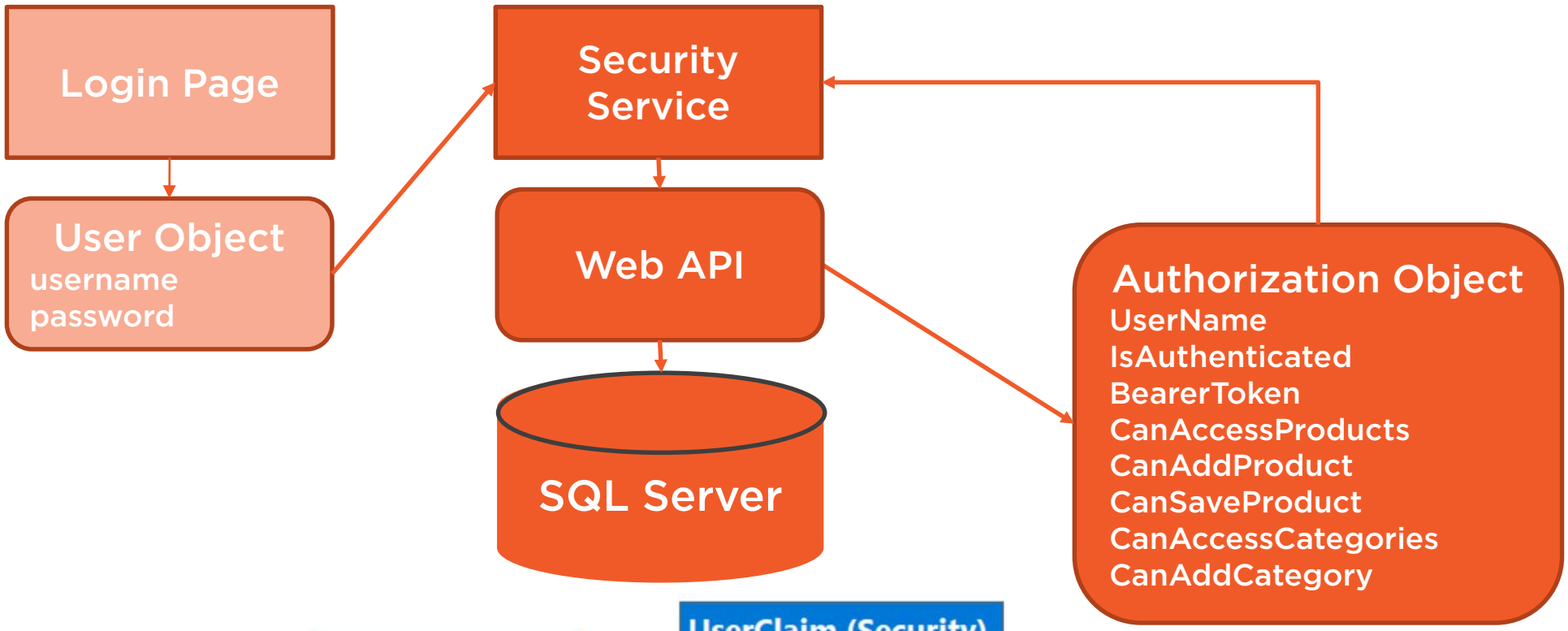
BUSINESS SOLUTIONS ARCHITECT, FAIRWAY TECHNOLOGIES, INC.

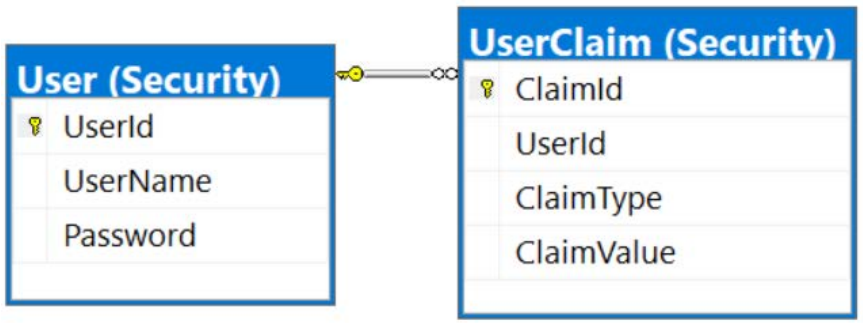
www.fairwaytech.com psheriff@fairwaytech.com

Goals

- SQL Server security tables
- Entity Framework classes
- Application user authorization class
- Security manager class
- Security controller
- Authenticate from Angular to Web API

Security Architecture Using Web API





Security Tables and Classes

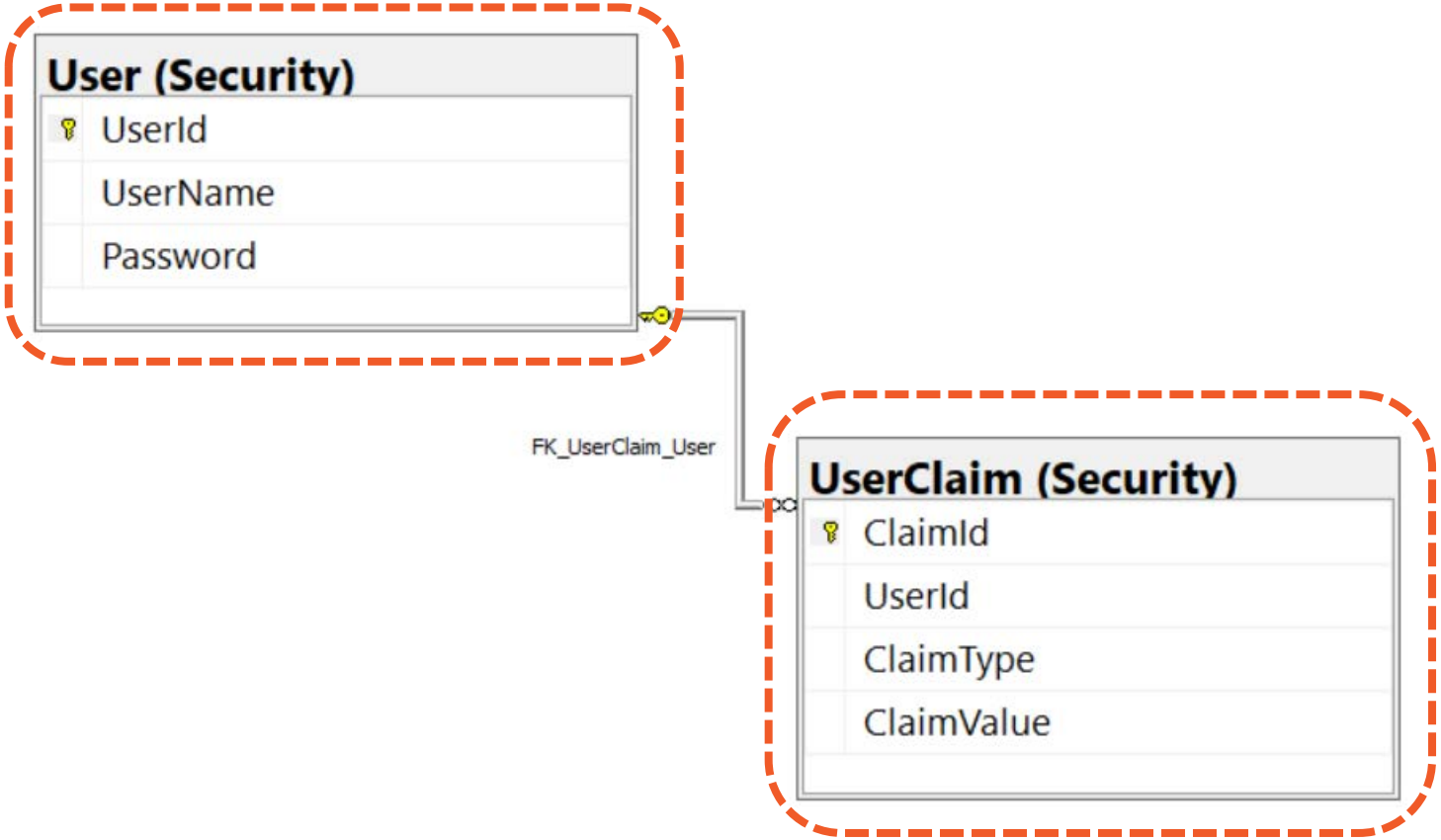
Security tables

User

UserClaim

Similar to ASP.NET Identity tables

Each user has a set of claims

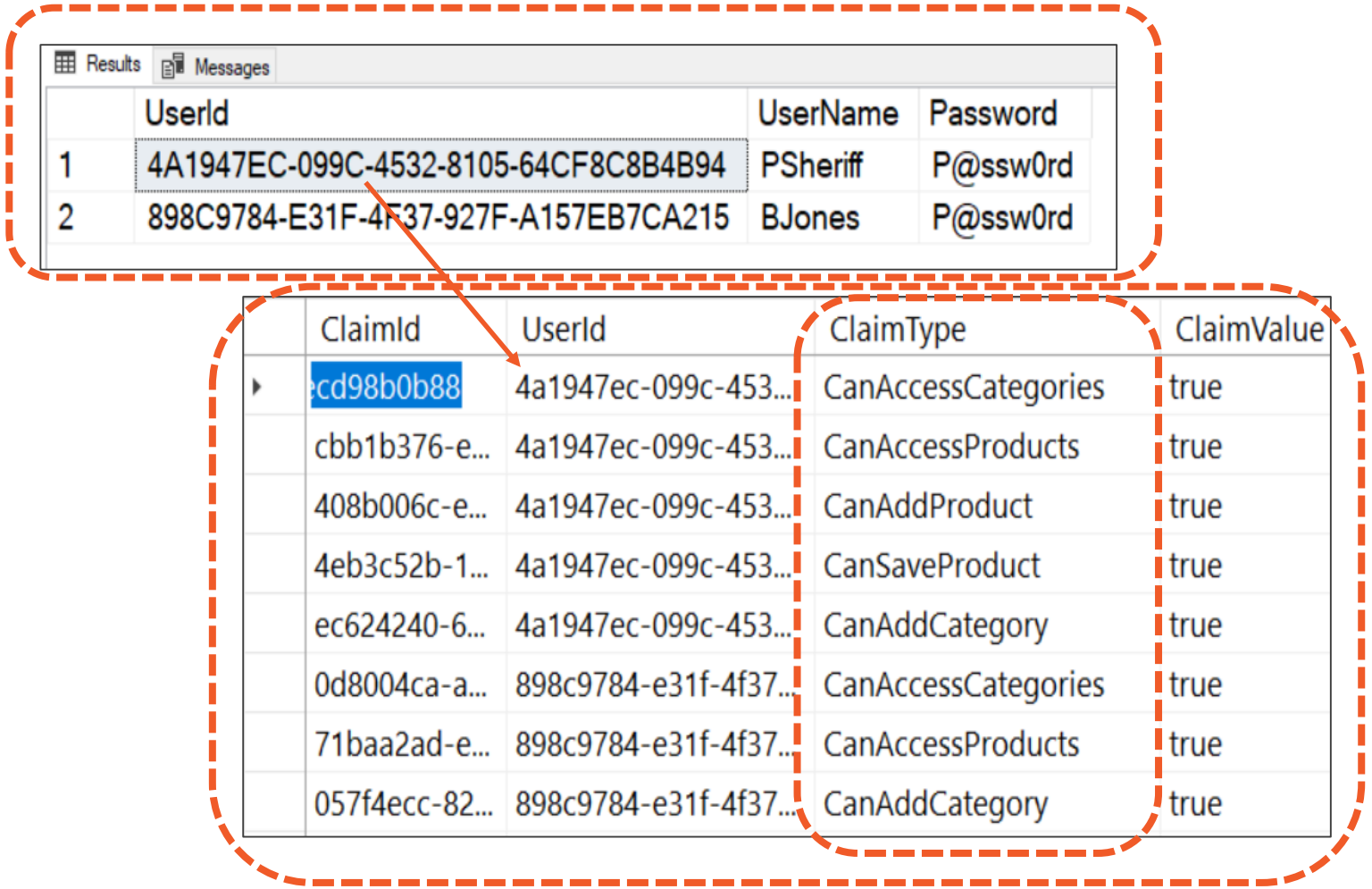


User table

User information for login

UserClaim table

ClaimType value matches property names in AppUserAuth class



AppUser class

Connect to user table

Add appropriate data annotations

```
[Table("User", Schema = "Security")]
6 references
public partial class AppUser
{
    [Required()]
    [Key()]
    1 reference
    public Guid UserId { get; set; }

    [Required()]
    [StringLength(255)]
    3 references
    public string UserName { get; set; }

    [Required()]
    [StringLength(255)]
    2 references
    public string Password { get; set; }
}
```

AppUserClaim class

Connect to user claim table

Add appropriate data annotations

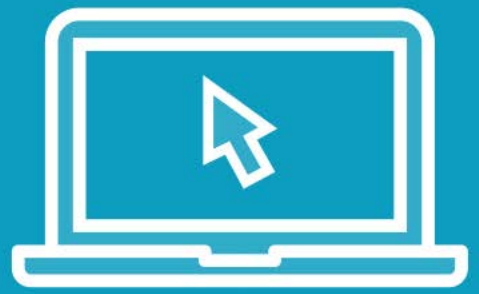
```
[Table("UserClaim", Schema = "Security")]
7 references
public class AppUserClaim
{
    [Required()]
    [Key()]
    0 references
    public Guid ClaimId { get; set; }

    [Required()]
    1 reference
    public Guid UserId { get; set; }

    [Required()]
    1 reference
    public string ClaimType { get; set; }

    [Required()]
    1 reference
    public string ClaimValue { get; set; }
}
```

Demo



Create Entity Framework security classes

Security Manager Class

AppUserAuth class

User info properties

User name

Bearer token

Is authenticated

UI control properties

Match names in Angular class

```
public class AppUserAuth
{
    public AppUserAuth() : base()
    {
        UserName = "Not authorized";
        BearerToken = string.Empty;
    }

    public string UserName { get; set; }
    public string BearerToken { get; set; }
    public bool IsAuthenticated { get; set; }

    public bool CanAccessProducts { get; set; }
    public bool CanAddProduct { get; set; }
    public bool CanSaveProduct { get; set; }
    public bool CanAccessCategories { get; set; }
    public bool CanAddCategory { get; set; }
}
```

Authorization Class

Map claim type in table to property of the same name

ClaimType	ClaimValue
CanAccessProducts	true
CanAddProduct	true
CanSaveProduct	true
CanAccessCategories	true
CanAddCategory	true

```
public class AppUserAuth
{

  // OTHER PROPERTIES HERE

  public bool CanAccessProducts { get; set; }
  public bool CanAddProduct { get; set; }
  public bool CanSaveProduct { get; set; }
  public bool CanAccessCategories { get; set; }
  public bool CanAddCategory { get; set; }
}
```

Security Manager class

ValidateUser method

Validate user name and password

Returns authorization object

GetUserClaims method

BuildUserAuth method

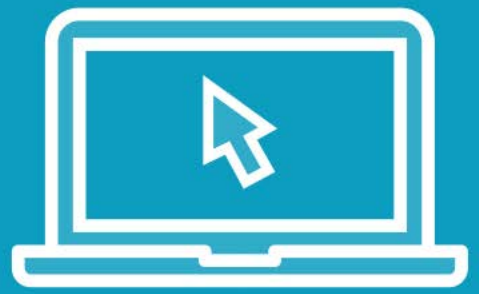
Map claims to authorization object

```
public class SecurityManager
{
    1 reference
    public AppUserAuth ValidateUser(AppUser user)
    {
    }

    1 reference
    protected List<AppUserClaim> GetUserClaims(AppUser authUser)
    {
    }

    1 reference
    protected AppUserAuth BuildUserAuthObject(AppUser authUser)
    {
    }
}
```

Demo



Authorization class

Security manager class

Security Controller

Security controller class

Login method

Calls security manager to authenticate

Returns status of 200

Payload is an AppUserAuth object

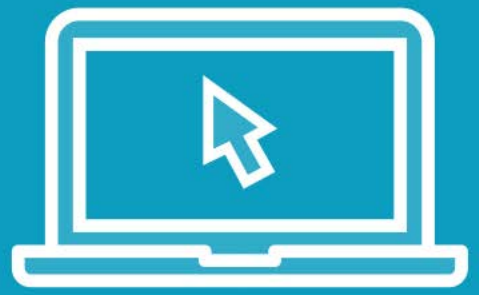
Otherwise returns status of 404

```
[Route("api/[controller]")]
0 references
public class SecurityController : Controller
{
  [HttpPost("login")]
  0 references
  public IActionResult Login([FromBody]AppUser user)
  {
    IActionResult ret = null;
    AppUserAuth auth = new AppUserAuth();
    SecurityManager mgr = new SecurityManager();

    auth = mgr.ValidateUser(user);
    if (auth.IsAuthenticated)
    {
      ret = StatusCode(StatusCodes.Status200OK, auth);
    }
    else
    {
      ret = StatusCode(StatusCodes.Status404NotFound,
                       "Invalid User Name/Password.");
    }

    return ret;
  }
}
```

Demo



Build security controller

Call Web API

Modify security.service.ts

Constant to point to security controller

Constant to post JSON login information

```
const API_URL = "http://localhost:5000/api/security/";

const httpOptions = {
  headers: new HttpHeaders({
    'Content-Type': 'application/json'
  })
};
```

Inject HttpClient into SecurityService class

```
@Injectable()
export class SecurityService {
  securityObject: AppUserAuth = new AppUserAuth();

  constructor(private http: HttpClient) { }
```

Modify login() method

Call Web API using http.post()

Pass AppUser object and *httpOptions* constant

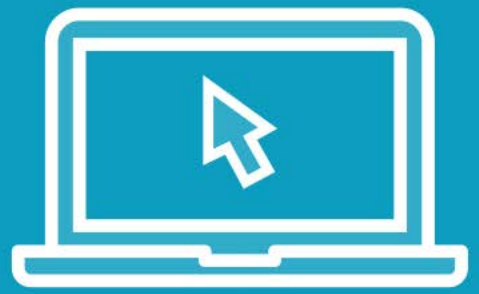
Assign back into *securityObject* property

Store bearer token into local storage

```
login(entity: AppUser): Observable<AppUserAuth> {
  // Initialize security object
  this.resetSecurityObject();

  return this.http.post<AppUserAuth>(API_URL + "login",
    entity, httpOptions).pipe(
      tap(resp => {
        Object.assign(this.securityObject, resp);
        // Store into local storage
        localStorage.setItem("bearerToken",
          this.securityObject.bearerToken);
      }));
}
```

Demo



Call Web API

Summary

- Authenticated the user using a Web API call instead of local mock data
- Returned authorization object from Web API
- Saw how to use reflection to map claims to properties

Coming up in the next module...

Learn to use JWT

What are bearer tokens?

How to generate a token using JWT