#### [Consuming REST Service in Angular with JWT](javascript:void(0);)

**Objectives**

* Demonstrate integration of RESTful Web Service with front end technologies
  + Base64 conversion in JavaScript, sending basic http authentication in headers, using token across components using service, verifying response HTTP status code, settings CORS in Spring Security, setting JWT in header, benefit of using HttpInterceptor
    - HttpHeader - https://angular.io/api/common/http/HttpHeaders
    - HttpClient Examples - https://blog.angular-university.io/angular-http/
    - Http Interceptor examples - https://blog.angulartraining.com/http-interceptors-in-angular-61dcf80b6bdd
    - Http Interceptor examples - https://scotch.io/@vigneshsithirai/angular-6-7-http-client-interceptor-with-error-handling

NOTE: There is no Quiz for this session

**Authenticate from Angular and get JWT**   
  
To integrate Login component with "/authentication" service and get the JWT follow steps below.  
  
**authentication.service.ts**

* Open the angular-learning project in Visual Studio Code
* Open the terminal window within Visual Studio Code
* For creating authentication service, execute the command below. This will create the authentication service in the login component folder.

ng generate service login/authentication

* Include a properties in authentication.service.ts, for storing the authentication REST Web Service URL and token. Refer code below:

  private authenticationApiUrl = 'http://localhost:8090/authenticate';

  private token: string;

* Include constructor with injection of HttpClient
* Include authenticate method that will be called from the login component for authenticating a user. Refer method signature below:

authenticate(user: string, password: string): Observable<any>

* In authenticate method concatenate user and password separated by colon
* Convert this concatenated string into base64 encoding using the btoa() method

btoa(user + ':' + password)

* Include HttpHeader with authencation info

    let headers = new HttpHeaders();

    headers = headers.set('Authorization', 'Basic ' + credentials);

* Invoke the get method of httpClient by passing the url and headers. Return the resultant header observable for the method.

this.httpClient.get(this.authenticationApiUrl, {headers})

* Include getter and setter method for token.

  public setToken(token: string) {

    this.token = token;

  }

  public getToken() {

    return this.token;

  }

**login.component.ts**

* Inject AuthenticationService in constructor
* Include error property for handling the error during login
* Update onSubmit() method to call authenticate method passing this.user and this.password
* Invoke the subscribe method on the Observable returned by the authentication method
* In the success part of the subscribe method, necessary aspects that needs to be done for successful login, the existing code that was present in this method can be reused here.
* In the success part invoke the authenticationService.setToken(() method passing data.token
* In the error part, check if the status code is 401 and set the error property value as "Invalid Username or Password"

**login.component.html**

* Include a placeholder for displaying error message with appropriate display login

**Testing**

* The invocation of REST Web service will fail with CORS error in the console
* To overcome CORS error inlucde the following line as the first line in the second configure method of SecurityConfig.java, which will enable CORS for Spring Security Framework.

httpSecurity.cors();

* Test the service from angular after inclusion of the above code.

**Authorization of Web Service invocation from Angular with JWT**   
  
Get all employees by passing the JWT token.  
  
Make following changes in employee.service.ts to invoke the RESTful Web Service call with token:

* Move the httpOptions constant definition to getAllEmployees() method
* Inject the AuthenticationService in the constructor
* Along with Content-Type setting add 'Authorization' header with values as 'Bearer ' + authenticationService.getToken()
* Try accessing the service and check if the update is invoke appropriately and the data gets modified

**Incorporate of all examples with JWT**   
  
Modify all the following features related to employee to work with JWT.

* Show edit employee form
* Edit Employee
* Delete Employee

The changes should reflect in the service and the changes in data should reflect appropriately

**Http Interceptor**   
  
By the time you had implemented the JWT based authentication for all employee related web service call, one could notice that the defintion of header is repeated everytime before invocationf the web service. To avoid this code duplication Http Interceptor concept in Angular can be incorporated.  
  
This is an additional activity after completion of JWT. This can be done if there is additional time available.  
  
Divide the class into multiple teams, each team having 3 to 4 members. This team members can be randomized or based on preference or by retaining existing seats.  
  
Each team has to implement Http Interceptor on the following two aspects:

* Inclusion of JWT token should be done using interceptor and the header inclusion in httpClient call should be removed
* Inclusion of error handling of all httpClient calls on the following aspect:
  + If error status code is 401, set the error message as 'Authentication Failed'
  + If error status code is 400, set the error message by getting the message from the error response
  + If the error status code is not 400 or 401, display the error message as 'System Error. Please retry or contact administrator.'

References for implementing interceptor. Feel free to find other relevant articles from internet.

* <https://scotch.io/@vigneshsithirai/angular-6-7-http-client-interceptor-with-error-handling>
* <https://blog.angularindepth.com/top-10-ways-to-use-interceptors-in-angular-db450f8a62d6> (Refer section 7 and 5)
* <https://blog.angulartraining.com/http-interceptors-in-angular-61dcf80b6bdd>