=================
Spring Security
=================

-> Security is very important for every web application.

-> To protect our application & application data we need to implement security logic.

-> Spring Security is one of the module of spring framework

-> Spring Security concept we can use to secure our web applications / REST APIs.

-> To implment security, we need to know about two concepts

1) Authentication

2) Authorization

-> Authentication means verifying who can access our application.

-> Authorization means verifying which user can access which functionality.

==============================
Working with Spring Security
==============================

-> To secure our spring boot application we need to add below starter in pom.xml file

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

Note-1: When we add this dependency in pom.xml file then by default our application will be secured with "http basic authentication".

Note-2: It will generate random password to access our application.

-> We need to use below credentials to access our application

```
                                Username : user

                                Password : <copy the pwd from console>
```

-> When we access our application url in browser then it will display "Login Form" to authenticate our request.

-> To access secured REST API from postman, we need to set Auth values in POSTMAN to send the request.

```
                                Auth : Basic Auth
                            Username : user
                                Password : <copy-from-console>
```

===
How to override Spring Security Default Credentials
===

-> To override Default credentials we can configre security credentials in "application.properties" file or "application.yml" file like below

```
spring.security.user.name=ashokit
spring.security.user.password=ashokit@123
```

=====================================
How to secure specific URL Patterns
=====================================

-> When we add 'security-starter' in pom.xml then it will apply security filter for all the HTTP methods of our application.

-> But in reality we need to secure only few methods not all methods in our application.

/login-page : secuirty not required (anyone can access)

/transfer : secuirty required

/balance : security required

/about-us : security not required

/contact-us : security not required

-> In order to achieve above requirement we need to Customize Security Configuration in our project like below.

```
@Configuration
@EnableWebSecurity
public class AppSecurityConfigurer {

        @Bean
        public SecurityFilterChain securityConfig(HttpSecurity http) throws Exception {

                        http.authorizeHttpRequests((req) -> req
                                        .requestMatchers("/welcome").permitAll()
                                        .anyRequest().authenticated()
                                ).httpBasic(Customizer.withDefaults())
                                 .formLogin(Customizer.withDefaults());

                        return http.build();
        }
}
```

==
Spring Security In-Memory Authentication
==

-> In Memory Authentication means storing user credentials in the program for Authentication Purpose.

Note: This is used only for practice purpose, not recommended in real-time.

```
@Bean
public InMemoryUserDetailsManager inMemoryUsers() {

        UserDetails u1 = User.withDefaultPasswordEncoder()
                                                 .username("ashokit")
                                                 .password("ashokit@123")
                                                 .build();

        UserDetails u2 = User.withDefaultPasswordEncoder()
                                                 .username("raju")
                                                 .password("raju@123")
                                                 .build();

        UserDetails u3 = User.withDefaultPasswordEncoder()
                                                 .username("john")
```

```
                                                    .password("john@123")
                                                    .build();

        return new InMemoryUserDetailsManager(u1, u2, u3);
}
```

===

Requirement-1 : Develop REST API with Http Basic Authentication and configure auth credentials in application.properties file or use in-memory authentication.

Note: Test this rest api from browser and from postman.

Requirement-2 : Develop Consumer application to accces above rest api (secured).

==

=> To access secured rest api we need to send basic auth credentials in request header like below

############ Authorization = Basic Base64.encode(uname:pwd) ############

==
Rest Template with Basic Authentication Header
==

```
String cred = uname+":"+pwd;

byte[] encodedCredentials = java.util.Base64.getEncoder().encode(cred);

String headerKey = "Authorization";
String headerValue = "Basic "+ new String(encodedCredentials);

HttpHeaders headers = new HttpHeaders();
headers.set(headerKey, headerValue);

HttpEntity entity = new HttpEntity(headers);

ResponseEntity<String> res =
                restTemplate.exchange(apiUrl, HttpMethod.GET, entity, String.class);

                String body = res.getBody();

                s.o.p(body);
```

==
WebClient with Basic Authentication Header
==

```
byte[] cred = java.util.Base64.getEncoder().encode(cred);

WebClient client = WebClient.create();

String response =  client.get( )
                                             .uri(apiUrl)
                                             .header("Authorization", "Basic "+ new String(cred))
                                             .retrieve( )
                                             .bodyToMono(String.class)
                                             .block( );

s.o.p(response);
```

==
Login and Registration using Spring Security
==

=> Develop springboot rest api with below 2 functionalities using Spring Security.

1) User Registration (name, email, pwd and phno)

2) User Login (email, pwd)

Note-1: When user register, we need to store user data in database table by encrypting user pwd.

Note-2: When user try to login, if credentials are valid send welcome msg as response. If credentials are invalid then send "Invalid Credential" msg as response.

Git Repo :: https://github.com/ashokitschool/springboot_register_login_security.git

=====================
Development Process
=====================

1) Create Boot app with required dependencies

a) web-starter
b) data-jpa-starter
c) mysql
d) security-starter
e) devtools

2) Configure Data Source properties in application.properties file

3) Create Entity class & Repository interface

4) Create CustomerService class by implementing UserDetailsService class

5) Create Security Config Class

6) Create RestController with required methods

7) Run the application and test it

```
{
    "name": "Sunil",
    "phno" : 6686868,
    "email" : "sunil@gmail.com",
    "pwd" : "sunil@1233"
}
```

==============================
Spring Boot with OAuth 2.0
==============================

1) Create oAuth app in github.com

(Login --> Profile -> Settings --> Developer Settings --> OAuth Apps --> Create App --> Copy Client ID & Client Secret)

Client ID :

Client Secret :

2) Create Spring Boot application with below dependencies

a) web-starter

```
                b) security-starter
                c) oauth-client
```

3)Create Rest Controller with method

```java
@RestController
public class WelcomeRestController {

        @GetMapping("/")
        public String welcome() {
                return "Welcome to Ashok IT";
        }
}
```

4) Configure GitHub OAuth App client id & client secret in application.yml file like below

```yaml
spring:
  security:
    oauth2:
      client:
        registration:
          github:
            clientId:
            clientSecret:
```

5) Run the application and test it.

===

Assignment : Spring Boot with oAuth using google account. Get username also from google and display that in response.

===

```
======================
Spring Boot with JWT
======================
```

-> JWT stands for JSON Web Tokens.

-> JWT official Website : https://jwt.io/

-> Below is the sample JWT Token

```
token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0Ij
oxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```

-> JWT contains below 3 parts

```
                1) Header
                2) Payload
                3) Signature
```

Note: JWT 3 parts will be seperated by using dot(.)

Note: Client application should send JWT Token to provider in below format

```
Authorization=Bearer
eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhbmlsQGdtYWlsLmNvbSIsImlhdCI6MTcyOTA1NzcxNiwiZXhwIjoxNzI5MDYxMzE2fQ.S
J1yonGgN85h7MewbsygPS8pN2JSHRn-6ICJ7bJVbvQ
```

Git Hub Repo : https://github.com/ashokitschool/SpringBoot_JWT_App.git

1) JWT Token generation (JwtService.java)

    - generateToken(String uname)
    - validateToken(String uname)

2) JWT Token validation Filter (AppFilter.java) - OncePerRequest

    - check Authorization header presence
    - retrieve bearer token from header
    - validate token
    - if token is valid, update security context to process req

3) Customize SecurityFilterChain

    - permit /api/register & /api/login urls
    - authenticate any other request

================================

Microservices with JWT Security

================================

Git Hub Repo : https://github.com/ashokitschool/Microservices_Security.git