```
Topics
  a. SpringSecurity(All levels of Authentication and authorization:: 3 classes)
  b. Junit(1 class)
  c. HttpUnit with Mockito(1 class)
  d. React and SpringBoot Integration(2 classes)
  e. JDBC app development using type-1,type-2,type-3 driver(deprecated: one class)
  e. Docker and K8s ( by other trainer -> Already started.)

SpringBoot Modules
      a.SpringCore
      b.SpringDataJPA
      c.SpringMVC
      d.SpringAOP
      e.SpringRest
      f.SpringJMS(SpringMail)
      g.SpringSecurity


SpringSecurity
   -> Pre-requisite :
CoreJava,Servlet,JSP,SpringCore,SpringDataJPA,SpringMVC,SpringRest.

SpringBoot Security
  => It is a Spring extension module that provides  mulitple ready made filter to
enable security on SpringBoot MVC and SpringRest applications.

Note:
 If we make filter component as a SpringBean IOC container then any othe dependent
spring bean object can be injected to that Servlet Filter
 SpringBean,but that servlet filter cannot trap the request from the browser
window.
 For this we need configure special ready made servlet filter component given by
SpringSecurity module called "DelegationFilterProxy" having
 URL Pattern "/",with logical name matching with ServletFilter component bean id.

How to Secure our RestAPI's using SpringBoot?
  => Security is most important for webapplication.
  => To protect our application and application data we need to implement security.
  => To provide security for webapplication and restful api's spring framework had
given a seperate module called "Spring-Security".

Create a SpringRest project with the following dependancies
      a. SpringWeb
      b. SpringSecurity
      c. DevTools

 To secure our Restapi's we need to add the following dependancy in pom.xml file
      <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
      </dependency>
 => When we add this dependancy our application will be secured with "Basic
Authentication".
 => To access the endpoints we need to login to the application with the following
credentials
            username : user
            password : <copy from the console>

+++++++++++++++++++++++
BankRestController.java
+++++++++++++++++++++++
package in.ineuron.nitin.restcontroller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class BankRestController {

      @GetMapping("/")
      public String welcomePage() {
            return "Welcome to ICICI Bank";
      }
}

browser
+++++++
request  : GET -> http://localhost:9999/
response : It will be redirected to http://localhost:9999/login
                  username : user
                  password : 95e4ce5c-9bdc-4909-ac3e-cb1985b30ce5(copied from
console after running the application)
           Welcome to ICICI Bank

POSTMAN
+++++++
 request  : GET -> http://localhost:9999/bank/
 response : It generates the response as 401(Unauthorized access)
            Go to authorization model in postman
                  username :  user
                  password :  95e4ce5c-9bdc-4909-ac3e-cb1985b30ce5(copied from
console after running the application)
            click on send button

 response : Welcome to ICICI Bank

Note:
 Q> The default password generated by the application is not userfriendly, if we
want to override the credentials it is possible?
answer: yes it is possible, to do so we need to use "application.properties".

application.properties
server.port = 9999
spring.security.user.name=root
spring.security.user.password=root123

Now if we start the application, SpringRest will not genreate the default password.

browser
+++++++
request  : GET -> http://localhost:9999/bank/
response : It will be redirected to http://localhost:9999/login
                  username : root
                  password : root123
           Welcome to ICICI Bank

POSTMAN
+++++++
 request  : GET -> http://localhost:9999/bank/
 response : It will be redirected to http://localhost:9999/login
            Go to authorization model in postman
                  username : root
                  password : root123
            click on send button
 response : Welcome to ICICI Bank(200 ok)

 response : It will be redirected to http://localhost:9999/login
            Go to authorization model in postman
                  username : root
                  password : root123#113
            click on send button
 response : 401(unAuthorized Access)
```

```
++++++++++++++++++++++++++++++++++++++++++++++
Restricting the authroization for few ENDPOINTS
++++++++++++++++++++++++++++++++++++++++++++++
@GetMapping("/transfer")
public String fundTransfer() {
            return "Fund transfer initiated";
}
@GetMapping("/balance")
public String checkBalance() {
      return "Balance amount is :: 10000INR";
}
@GetMapping("/about")
public String aboutUs() {
      return "ICICI bank is managed by India Central Govt";
}

request : http://localhost:9999/bank/transfer/
request : http://localhost:9999/bank/balance/
request : http://localhost:9999/bank/about/
```

All the endpoints mentioned above will be authenticated, but in realtime only few endpoints should be authenticated but not all.

```
      eg: SBI application
            /about us     -> not authentication
              /fundtransfer -> authentication is required(login is required)
            /balance      -> authentication is required(login is required)
```

In our application,by default all the endpoints are authenticated, if we want to authenticate few endpoints then we need to for customzation.
To secure our own URL patterns, we need to use "SecurityConfiguration" as shown below.

```
package in.ineuron.nitin.securityconfiguration;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.web.SecurityFilterChain;

@Configuration
@EnableWebSecurity
public class SecurityConfigApp {
      @Bean
      public SecurityFilterChain userDefinedFilter(HttpSecurity http) throws
Exception {
            http.authorizeHttpRequests(
                        request -> request.antMatchers("/bank/", "/login",
"/bank/about")
                        .permitAll()
                        .anyRequest()
                        .authenticated())
                        .formLogin();

            return http.build();
      }
}
output
request  : http://localhost:9999/bank/
response : Welcome to ICICI Bank

request  :http://loclahost:9999/bank/transfer
response :It will be redirected to http://localhost:9999/login
            Go to authorization model in postman
                  username : root
                  password : root123
            click on send button
 response : Fund transfer initiated

request  :http://loclahost:9999/bank/balance
response : It will be redirected to http://localhost:9999/login
            Go to authorization model in postman
                  username : root
                  password : root123
            click on send button
  response : Balance amount is :: 10000INR

request  : http://loclahost:9999/bank/about
response : ICICI bank is managed by India Central Govt
```

Note: If the credentials supplied by the user is wrong, then it will be redirected to "http://localhost:9999/login?error"[Bad Credentials]

In the above application, the credentials information is hardcoded inside application.properties file,but in reality the credentials information will be stored in the database. To do so we need to use "SpringSecurity with JDBC authentication".

+++++++++++++++++++++++++++++++++++++++
SpringSecurity with JDBC authentication
+++++++++++++++++++++++++++++++++++++++

1. Create a Springstarter project with the following dependancies
   a. SpringDev tools
   b. MySQLDriver
   c. Spring Data JDBC
   d. SpringSecurity
   e. SpringWeb

2. application.properties

```
    server.port = 9999
    spring.datasource.url =jdbc:mysql://localhost:3306/enterprisejava
    spring.datasource.username= root
    spring.datasource.password= root123
```

3. Create a REST endpoints as shown below

```
package in.ineuron.nitin.restcontroller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/api")
public class UserRestController {

      @GetMapping("/")
      public String welcome() {
            return "<h1>Welcome to iNeuron Family</h1>";
      }

      @GetMapping("/admin")
      public String adminProcess() {
            return "<h1>Welcome Admin</h1>";
      }

      @GetMapping("/user")
      public String userProcess() {
            return "<h1>Welcome User</h1>";
      }

}
```

4. Create a CustomConfigation for handling the RestEndPoints.

```
package in.ineuron.nitin.securitfyconfiguration;

import javax.sql.DataSource;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication.builders.Authenticati
onManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;

@Configuration
@EnableWebSecurity
public class SecurityConfigApp {

      @Autowired
      private DataSource dataSource;

      @Autowired
      public void authenticationManager(AuthenticationManagerBuilder auth) throws
Exception {
```

```
            auth
            .jdbcAuthentication()
            .passwordEncoder(new BCryptPasswordEncoder())
            .dataSource(dataSource)
            .usersByUsernameQuery("select username,password,enabled from users
where username =?")
            .authoritiesByUsernameQuery("select username,authority from authorities
where username=?");
      }

      @Bean
      public SecurityFilterChain customFilterChain(HttpSecurity http) throws
Exception {
            http.authorizeHttpRequests(
                        request -> request.antMatchers("/api/").permitAll()
                        .antMatchers("/api/admin/").hasRole("ADMIN")
                        .antMatchers("/api/user/").hasAnyRole("ADMIN","USER")
                        .anyRequest().authenticated()
                        ).formLogin();


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

Run the application and send the request

```
request  : http://localhost:9999/api/
response : Welcome to iNeuron Family

request  :http://localhost:9999/api/admin
response : It will redirected to http://localhost:9999/login
                  username : sachin
                  password : sachin@123

response : Welcome Admin

request  :http://localhost:9999/api/user
response : It will redirected to http://localhost:9999/user
                  username : gill
                  password : gill@123

response : Welcome User
```