A deep dive into the
Spring Security

Sepehr S. T. Co. Ltd.

Implementing Secure and Scalable Webservices

ECE Department of Shahid Beheshti University

Mohammad Mehdi Moghimi

Senior Software Developer

What I’m assuming

- You’re familiar with Java
- You’re at least somewhat with Spring/SpringBoot
- You can read Java doc for what I’m not covering
- You’re eager to Learn new thing

What is Spring Security

- Provides Enterprise-Level Authentication & Authorization Services
- Authentication Types:
  - Simple Form Based
  - HTTP Basic and Digest
  - LDAP
  - X. 509 Client Cert
  - OpenID
  - etc

Spring Security History

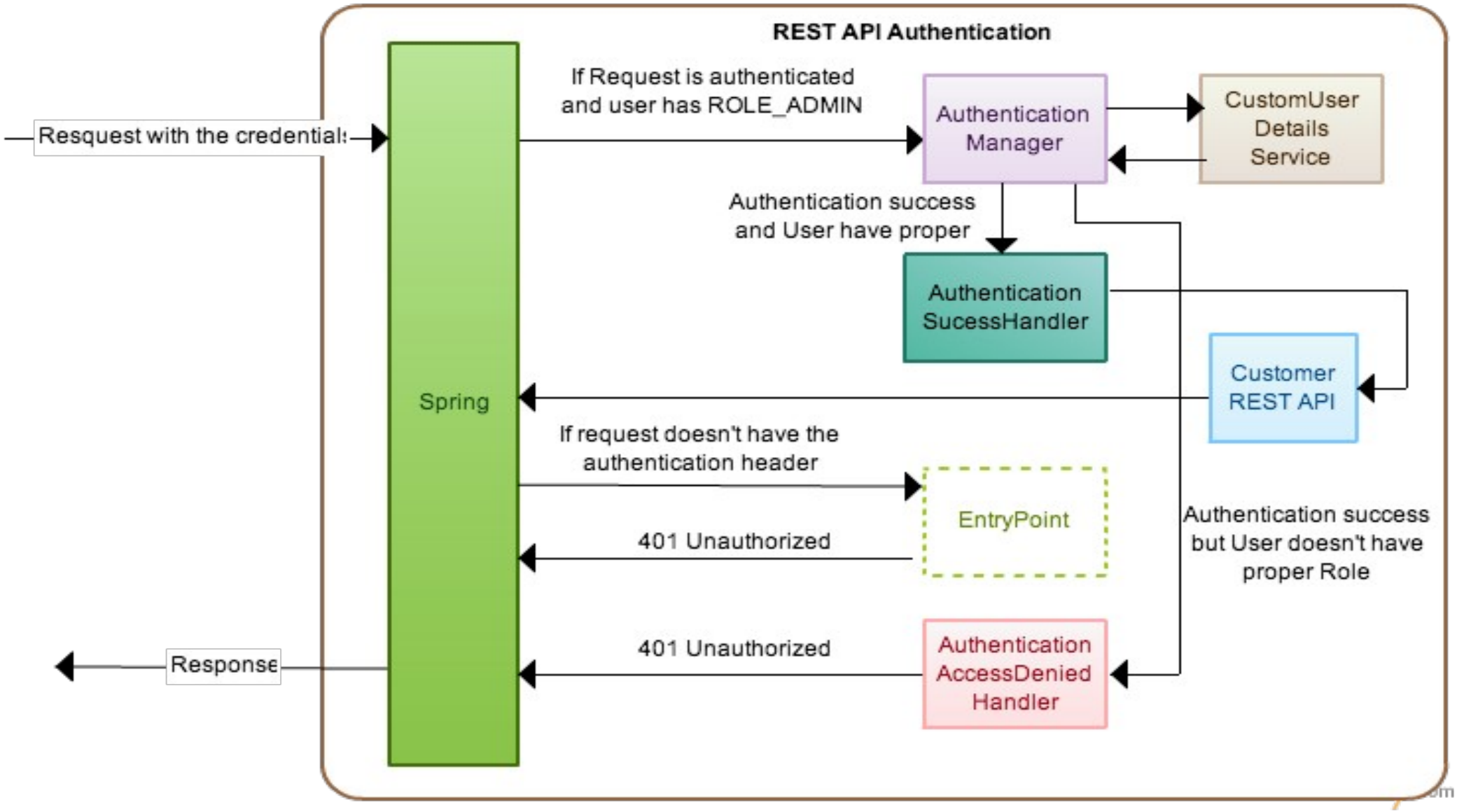
- ACEGI Project
- Rebranded as Spring Security in Spring 2.0
- Death By XML Configuration
- Configuration By Convention

Basic Concepts

- PRINCIPAL (Person Logging into the Application)
- `GrantedAuthority`
  - What permission Principal has
- `SecurityContext`
  - Holds the authentication
- `SecurityContextHolder`
  - Provides Access to `SecurityContext`

Basic Concept - 2

- `UserDetails`
  - Provides information to build an Authentication
- `UserDetailsService`
  - Creates a `UserDetails` from a passed String


REST API Authentication
Resquest with the credentials
Spring
If Request is authenticated and user has ROLE_ADMIN
Authentication Manager
CustomUser Details Service
Authentication success and User have proper
Authentication SucessHandler
Customer REST API
If request doesn't have the authentication header
EntryPoint
401 Unauthorized
Authentication success but User doesn't have proper Role
401 Unauthorized
Authentication AccessDenied Handler
Response

Lets dive into the code

```java
import com.sun.org.apache.xpath.internal.objects.XString;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@Controller
public class HomeController {

    @RequestMapping("/")
    public String homeController(){
        return "home.jsp";
    }
}
```

```jsp
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
    <title>SBU Login Form</title>
</head>
<body>
    <p> Hello Dear Students...!</p>
</body>
</html>
```

Dependencies

- Spring Security Web

```
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-web</artifactId>
    <version>3.1.0.RELEASE</version>
</dependency>
```

- Spring Security config

```
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-config</artifactId>
    <version>3.1.0.RELEASE</version>
</dependency>
```

Starter for SpringBoot

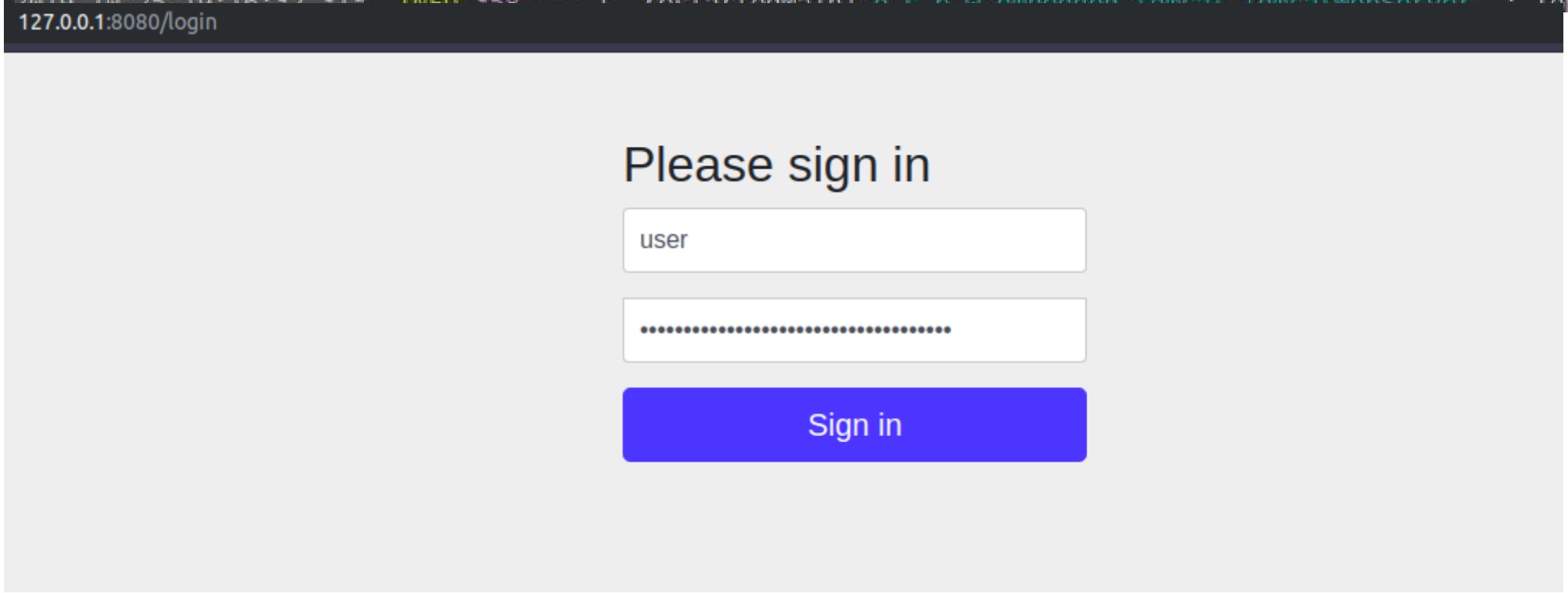
```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

See the awesome result

```
2019-10-25 19:16:32.125  INFO 358 --- [  restartedMain] .s.s.UserDetailsServiceAutoConfiguration :

Using generated security password: 0b258076-5a8d-40a2-961e-9a0844ff9818

2019-10-25 19:16:32.213  INFO 358 --- [  restartedMain] o.s.s.web.DefaultSecurityFilterChain     : Crea
2019-10-25 19:16:32.264  INFO 358 --- [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : Live
```



Hello Dear Students...!

Acceptable.. but not good enough!

- Why?
- Do you remember UserDetailService??

```
@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Bean
    @Override
    protected UserDetailsService userDetailsService() {
        List<UserDetails> userDetails= new ArrayList<>();
        userDetails.add(User.withDefaultPasswordEncoder().username("moghimi").password("123456").roles("USER").build());
        userDetails.add(User.withDefaultPasswordEncoder().username("farzad").password("456789").roles("Admin").build());

        return new InMemoryUserDetailsManager(userDetails);
    }
}
```

Lets make it more secure

- Dependencies for JPA and MySQL connector

```
<!--
https://mvnrepository.com/artifact/org.springframework.boo
t/spring-boot-starter-data-jpa -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
    <version>2.2.0.RELEASE</version>
</dependency>

<!-- https://mvnrepository.com/artifact/mysql/mysql-
connector-java -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.18</version>
</dependency>
```

Prerequisite

- Datasource configuration in application.properties

```
spring.datasource.url = jdbc:mysql://localhost:3306/sbu?useSSL=false
spring.datasource.username = root
spring.datasource.password = ******
spring.datasource.driver-class-name=com.mysql.
```

```
@Entity
public class User {
    @Id
    String username;
    String password;
    Boolean enabled;

    public User() {
    }

    public User(String username, String passwo
        this.username = username;
        this.password = password;
        this.enabled = enabled;
    }
```

```
@Repository
public interface UserRepository extends JpaRepository<User, String> {
    public User findUserByUsername(String username);
}
```

Need to have AuthenticationProvider

```
@Autowired
private MyUserDetailService userDetailsService;
 @Bean
 public AuthenticationProvider authenticationProvider(){ DaoAuthenticationProvider authenticationProvider =
         new DaoAuthenticationProvider();
    authenticationProvider.setUserDetailsService(userDetailsService);
    authenticationProvider.setPasswordEncoder(NoOpPasswordEncoder.getInstance());
    return authenticationProvider;

 }
```

```
rvice
ublic class MyUserDetailService implements  UserDetailsService {
    @Autowired
    UserRepository userRepository;
    @Override
    public UserDetails loadUserByUsername(String s) throws UsernameNotFoundException
        User user= userRepository.findUserByUsername(s);
        if(user==null){
            throw  new UsernameNotFoundException("user not found");
        }
        return new MyUserDetails(user);
    }
```

```
@Configuration
@Order(SecurityProperties.ACCESS_OVERRIDE_ORDER)
public class SecurityConfig extends WebSecurityConfigurerAdapter {

@Autowired
DataSource datasource;

@Override
protected void configure(HttpSecurity http) throws Exception {
    http
        .authorizeRequests()
            .anyRequest()
            .fullyAuthenticated()
            .and()
        .formLogin()
            .loginPage("/login")
            .failureUrl("/login?error")
            .permitAll()
            .and()
        .logout()
            .logoutUrl("/logout")
            .logoutSuccessUrl("/login?logout")
            .permitAll()
            .and()
        .csrf();
}

@Override
protected void configure(AuthenticationManagerBuilder auth) throws Exception {
    auth.jdbcAuthentication().dataSource(datasource).passwordEncoder(passwordEncoder());
}
```

```
@Override
protected void configure(AuthenticationManagerBuilder auth) throws Exception {
    auth.jdbcAuthentication().dataSource(datasource).passwordEncoder(passwordEncoder());
}

@Bean
public PasswordEncoder passwordEncoder() {
    PasswordEncoder encoder = new BCryptPasswordEncoder();
    return encoder;
```

Spring Security Looks for

```
create table users (
  username varchar(50) not null primary key,
  password varchar(255) not null,
  enabled boolean not null) ;

create table authorities (
  username varchar(50) not null,
  authority varchar(50) not null,
  foreign key (username) references users (username),
  unique index authorities_idx_1 (username, authority));
```

SPEL

Verb	Arguments	Description
hasRole(…)	Role Name	Permit access only users with the specified role
hasAnyRole(…)	Comma separated list of roles	Permit access only to users who have at least one of the roles specified in the comma separated list of roles
permitAll()	none	Give access to everybody
denyAll()	none	Give access to nobody
isAuthenticated()	none	Give access to all users who are authenticated. Deny access for request by users who have not been authenticated
isFullyAuthenticated()	none	Give access to users who have been authenticated by logging in. Deny access to users who have been authenticated by way of the remember me feature

Pattern Matching in SPEL

pattern	SpEL verb	Allows	Denies
/Static/*	permitAll()	/static/main.htm	/static/view/view1.htm
/static/**	permitAll()	/static/main.htm /static/view/view1.htm	
/static/* /static/**	permitAll() hasRole(...)	/static/main.htm	/static/view/view1.htm
/static/*.json /static/* /static/**	hasRole(...) permitAll() hasRole(…)	/static/main.htm	/static/catalog.json /static/view/view1.htm
/static/*.htm /static/**.js /static/**	permitAll() permitAll() denyAll()	/static/main.htm /static/main.js /static/view/view1.js	/static/main.jpg /static/view/view1.htm

Lets Code Together

- Implementing LDAP based AA in Spring Security

Exercise ;-)

- Add Security on top of our APIs based on OAUTH2