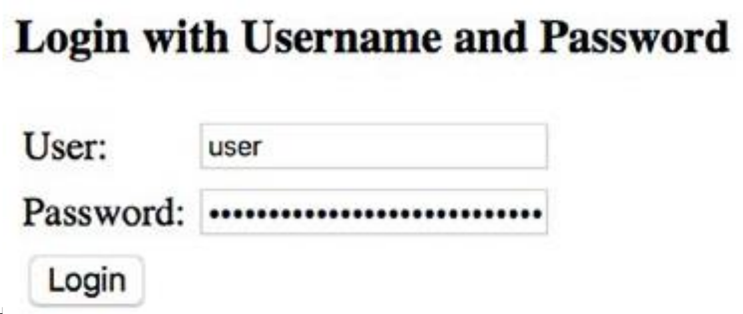
Authentication and Authorization

WebSecurityConfigurerAdapter

Security

- Security services
  - Confidentiality
  - Authentication
  - Authorization
- The first step is to add `spring-boot-starter-securi`
- Declarative security
  - spring.security.user.name=apress
  - spring.security.user.password=springboot2
  - spring.security.user.roles=ADMIN
- Programmatic security
  - Extend WebSecurityConfigureAdapter class


Login with Username and Password

User: user
Password: ••••••••••••••••••••••••••••

```
  /***https://spring.io/guides/gs/securing-web/*/
  /***https://docs.spring.io/spring-cloud-skipper/docs/1.0.0.BUILD-SNAPSHOT/reference/html/configuration-security-enabling-
  https.html*/
@Configuration          @EnableWebSecurity
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {
            @Override
            protected void configure(HttpSecurity http) throws Exception {
                        http
                                    .authorizeRequests()
                                                .antMatchers("/").permitAll()
                                                .anyRequest().authenticated()
                                                .and()
                                    .formLogin()
                                                //.loginPage("/login")
                                                .permitAll()
                                                .and()
                                    .logout()
                                                .permitAll()
                                                .and()
                                                .httpBasic();
                        http.csrf().disable();

            }
            @Bean
            @Override
            public UserDetailsService userDetailsService() {
                        UserDetails user = User.withDefaultPasswordEncoder().username("user")
            .password("password").roles("USER").build();
                        return new InMemoryUserDetailsManager(user);
```

The HttpSecurity class allows you to configure web-based security for specific HTTP requests. By default, it is applied to all requests, but can be restricted using requestMatcher(RequestMatcher) or similar methods

Settings for 8443

```
keytool -genkey -noprompt -alias tomcat-localhost -keyalg RSA -keystore
C:\Users\chand\localhost-rsa.jks -keypass 123456 -storepass 123456 -dname
"CN=tomcat-cert, OU=JU, O=JU, L=WB, ST=WB, C=IN“
```

```
<Connector
        protocol="org.apache.coyote.http11.Http11NioProtocol"
        port="8443" maxThreads="200"
        scheme="https" secure="true" SSLEnabled="true"
        keystoreFile="C:\my-cert-dir\localhost-rsa.jks"
        keystorePass="123456"
        clientAuth="false" sslProtocol="TLS"/>
```

Spring

- keytool -genkey -alias skipper -keyalg RSA -keystore c:\User\user1\skipper.keystore   -validity 3650 -storetype JKS -dname "CN=localhost, OU=Spring, O=Pivotal, L=Holualoa, ST=HI, C=IN" -keypass skipper -storepass skipper

❑ This method generates the key needed for HTTPS.
❑ Excute this command from jdk/bin of your machine
❑ Move the generated keystore file to the “resources” folder of your application

Confidentiality

Keep the following methods in the file where the main method is present

```
 @Bean
public ServletWebServerFactory servletContainer() {
TomcatServletWebServerFactory tomcat = new
TomcatServletWebServerFactory() {
@Override
 protected void postProcessContext(Context context) {
SecurityConstraint securityConstraint = new
SecurityConstraint();
securityConstraint.setUserConstraint("CONFIDENTIAL");
SecurityCollection collection = new SecurityCollection();
collection.addPattern("/*");
securityConstraint.addCollection(collection);
context.addConstraint(securityConstraint);            }
};
tomcat.addAdditionalTomcatConnectors(redirectConnector());
return tomcat;    }
```

Confidentiality

- Keep the following methods in the file where the main method is present

```
 private Connector redirectConnector() {
        Connector connector = new 
Connector("org.apache.coyote.http11.Http11NioProtocol");
        connector.setScheme("http");
        connector.setPort(8080);
        connector.setSecure(false);
        connector.setRedirectPort(8443);
        return connector;
    }
```