5. Securing Spring

# 5.1 Enabling Spring Security

-Add SB security starter dependency:



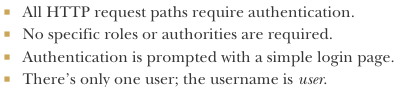
+This dependency is the only thing that’s required to secure an app.

-**Note**: incognito: have a fresh session each time opening a private/incognito window.

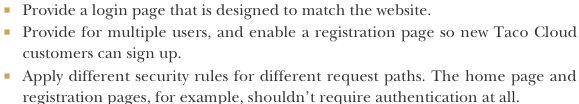
-You need to provide a username + password. User name is *user.* Password is randomly generated and written to app log file:



-Some security features when adding security starter:

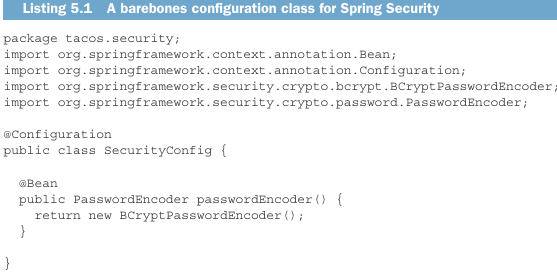


-We’ll configure Spring Security to do:



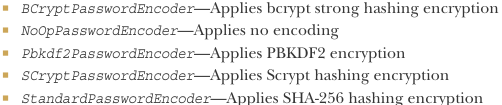
# 5.2 Configuring authentication

-Write the **configuration class**:



+It declares **PasswordEncoder** bean: use when create new users and authenticate users at login.

+**BCryptPasswordEncoder**: password encoders. Others:



+The password in the database must be never decoded. The password that the user enter at login is encoded, it’s then compared with the encoded password in database by PasswordEncoder **matches()**

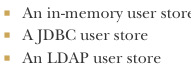
**-**Configure handle more than one user.

**-**Configure a user store for authentication: declare **UserDetailService** bean.



+**loadUserByUsername()**: use username to look up UserDetails object.

+SS offers some out-of-the-box implementation of UserDetailService

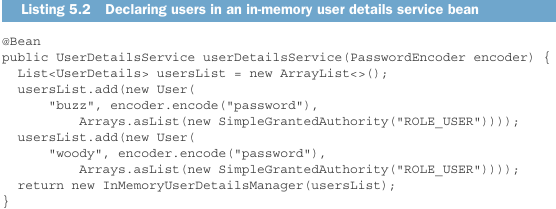


+Or you can create your own implementation.

## 5.2.1 In-memory user details service

-One place where user info can be kept in memory. None of users are likely to change may be defined as part of security configuration.

-Create InMemoryUserDetailManager bean method:



-**in-memory user details service** is **convenient** for **testing purposes** or for very simple apps, but it **doesn’t allow** to easy **edit users.**

## 5.2.2 Customizing user authentication

# 5.3 Securing web requests

## 5.3.1 Securing requests

## 5.3.2 Creating a custom login page

## 5.3.3 Enabling third-party authentication

## 5.3.4 Preventing cross-site request forgery

# 5.4 Applying method-level security

# 5.5 Knowing your user

# -Summary