**What is Spring Security?**

Spring Security is a powerful and highly customizable authentication and access-control framework. It is the de-facto standard for securing Spring-based applications.

Spring Security is a framework that focuses on providing both authentication and authorization to Java applications. Like all Spring projects, the real power of Spring Security is found in how easily it can be extended to meet custom requirements.

**Feature of Spring Security**

* Comprehensive and extensible support for both Authentication and Authorization
* Protection against attacks like session fixation, clickjacking, cross site request forgery, etc
* Servlet API integration
* Optional integration with Spring Web MVC.

**What is session fixation?**

* Session Fixation is an attack that permits an attacker to hijack a valid user session.

**What is clickjacking?**

* Clickjacking is a malicious technique of tricking a user into clicking on something different from what the user perceives, thus potentially revealing confidential information or allowing others to take control of their computer while clicking on seemingly innocuous objects,

including web pages.

Note: Think about that Spring security is just like a security guard, what they want to verify

* Who are you?
* What do you want?

**Why Spring Security?**

Application Security Framework

* Login and Logout Functionality
* Allow/block access to URLs to logged in users
* Allow/block access to URLs to logged in users AND with certain roles.

**What spring security can do?**

* user name/ password authentication
* sso/Okta/LDAP
* App Level Authorization
* Intra App Authorization like Oauth
* Microservice security(Using tokens, JWT)
* Method Level Security

**What is OKTA?**

* Okta is an identity access and management company that provides a whole host of software-as-service identity products. We have an implementation of OAuth 2.0 and OpenID Connect that makes adding single sign-on (SSO) to a Spring Boot app easy.

**5 core concepts in Spring Security**

1. Authentication

* Authentication is the process of determining whether someone or something is, in fact, who or what it declares itself to be.
* "Who are you?"

1. Authorization

* Authorization is the process of giving someone permission to do or have something.
* "Can this user do this?"

1. Principal

* The principal is just an Object. Most of the time this can be cast into a UserDetails object.
* Currently Logged in user

One User can have multiple IDS

But there is usually just one logged in user-> principal

(Per Request)

1. Granted Authority

* Represents an authority granted to an Authentication object. A GrantedAuthority must either represent itself as a String or be specifically supported by an AccessDecisionManager.

1. Roles

**Knowledge Based Authentication**

Knowledge-based authentication, commonly referred to as KBA, is a method of authentication which seeks to prove the identity of someone accessing a service such as a financial institution or website. As the name suggests, KBA requires the knowledge of private information of the individual to prove that the person providing the identity information is the owner of the identity. There are two types of KBA:

Static KBA, which is based on a pre-agreed set of shared secrets, and

Dynamic KBA, which is based on questions generated from a wider base of personal information.

Example of Knowledge Base KBA

Password, Pin Code, Answer to a secret/personal questions

**Advantage of Knowledge Based Authentication**

Easy to implement and use

**Possession Based Authentication**

Authentication mechanism based on what the user has, such as memory cards and smart card tokens. Possession-based authentication is also referred to as token-based authentication.

Examples: Phone/Text messages, Key cards and badges, Access Token device

**Multi Factor Authentication**

Multi-factor authentication (MFA) is an authentication method in which a computer user is granted access only after successfully presenting two or more pieces of evidence (or factors) to an authentication mechanism -knowledge (something the user and only the user knows), possession (something the user and only the user has), and inherence (something the user and only the user is).

Two-factor authentication (also known as 2FA) is a type, or subset, of multi-factor authentication. It is a method of confirming users' claimed identities by using a combination of two different factors:

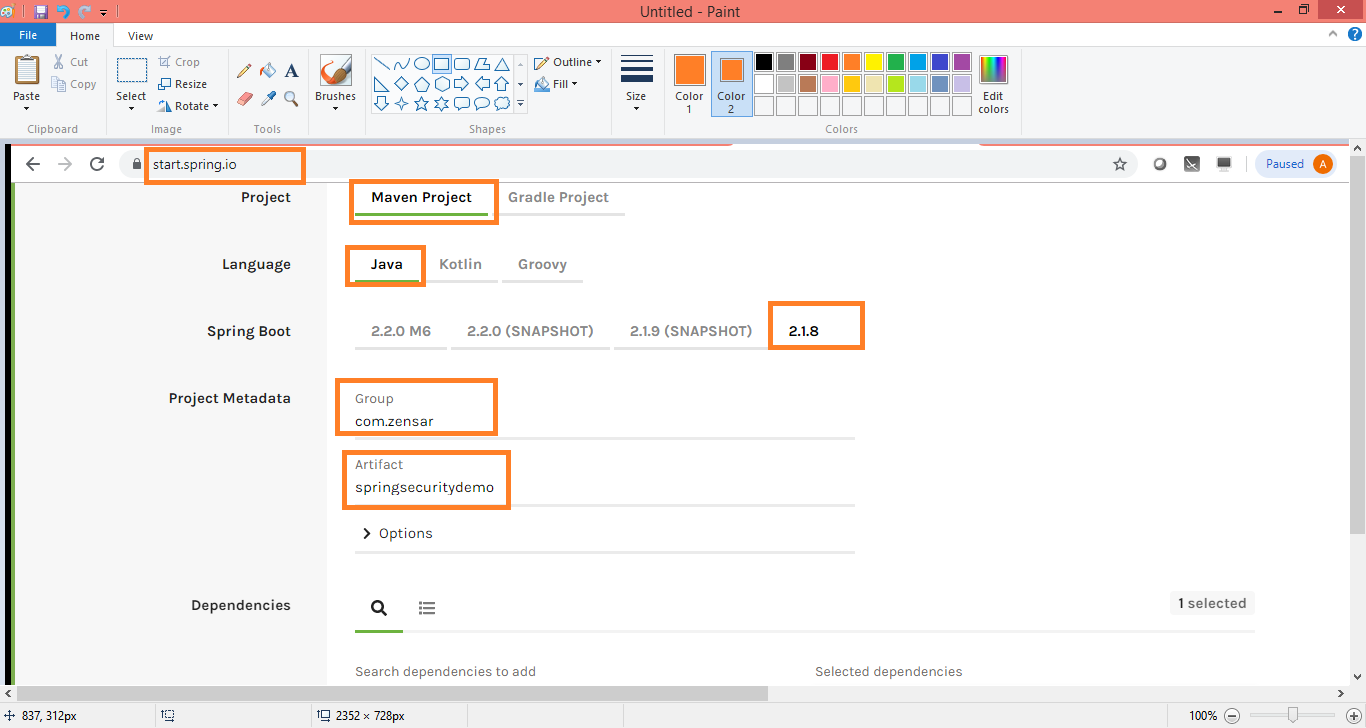
1) Something they know, 2) something they have, or 3) something they are.

A good example of two-factor authentication is the withdrawing of money from an ATM; only the correct combination of a bank card (something the user possesses) and a PIN (something the user knows) allows the transaction to be carried out.

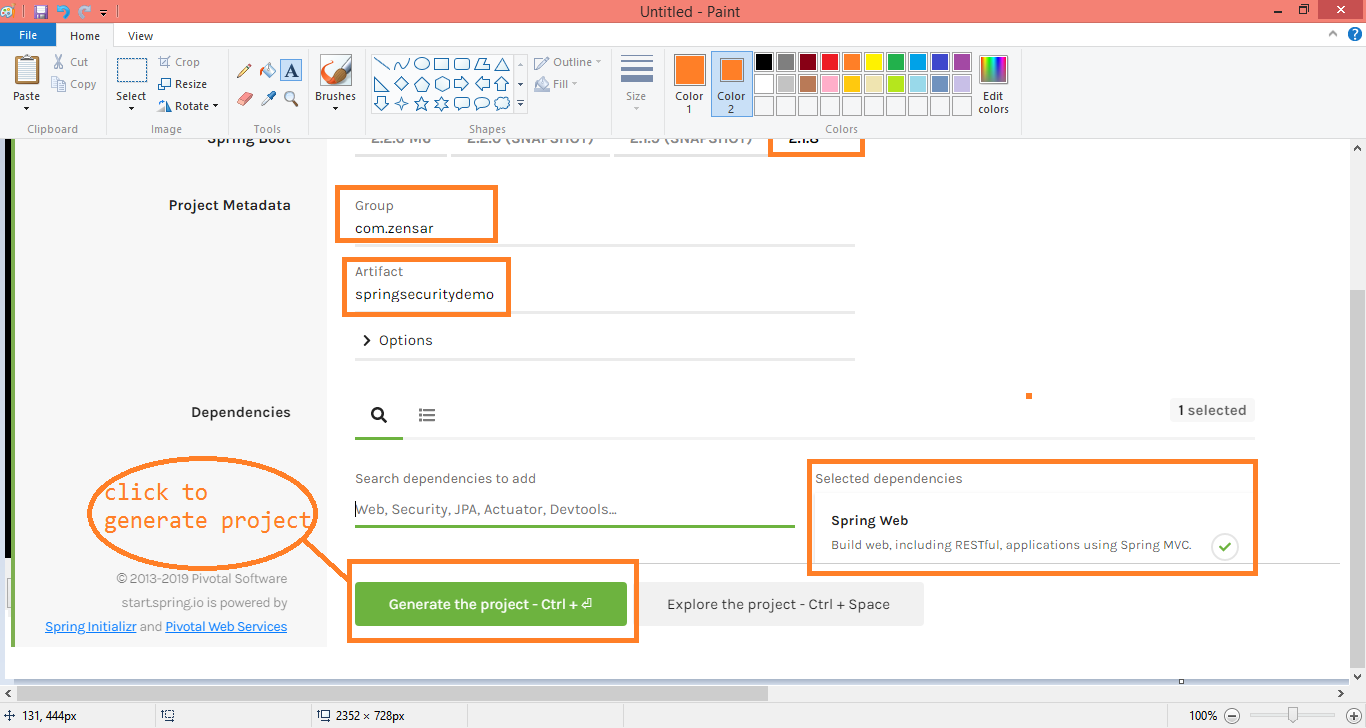
**How to add Spring Security to Spring Boot**

Goto start.spring.io

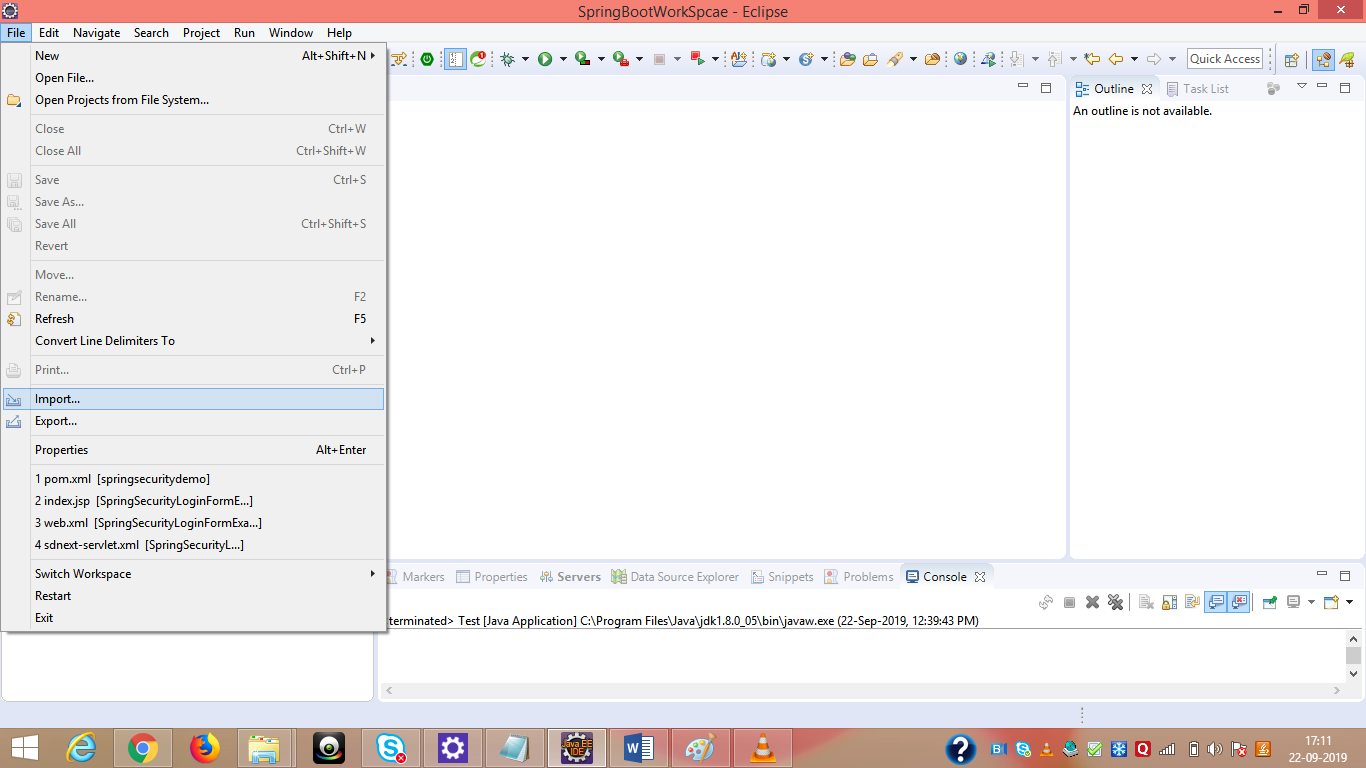
Select all as mention below

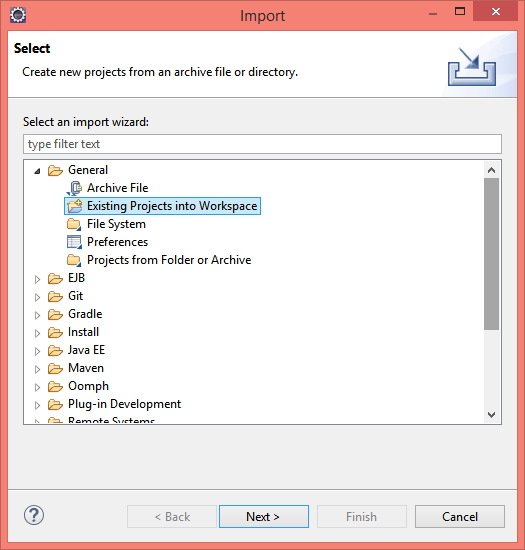


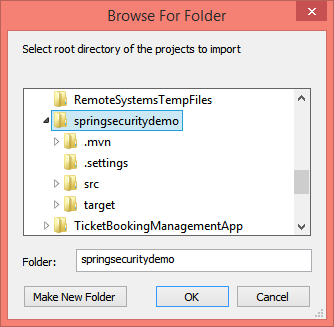
Generate Project

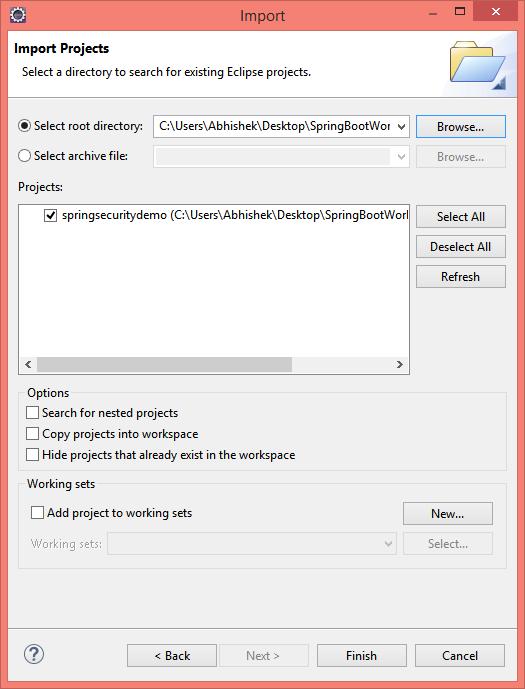


Import that project as Maven Project

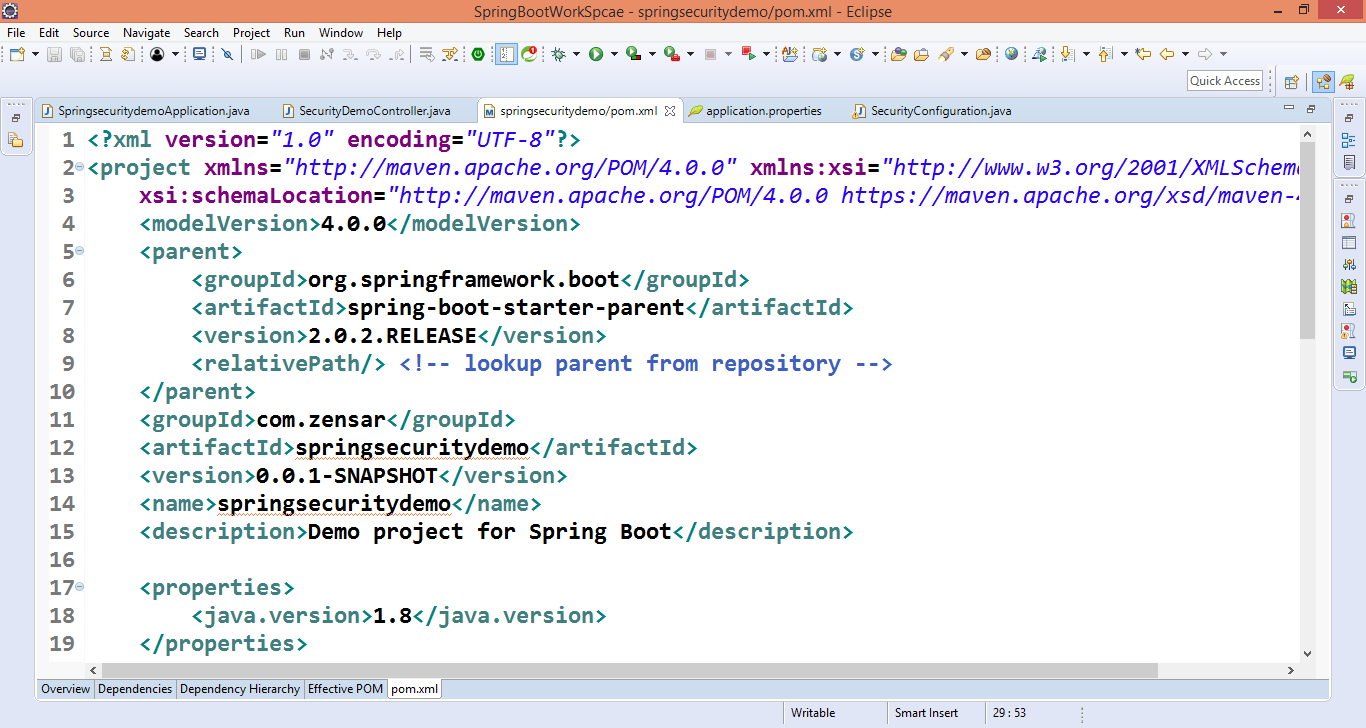




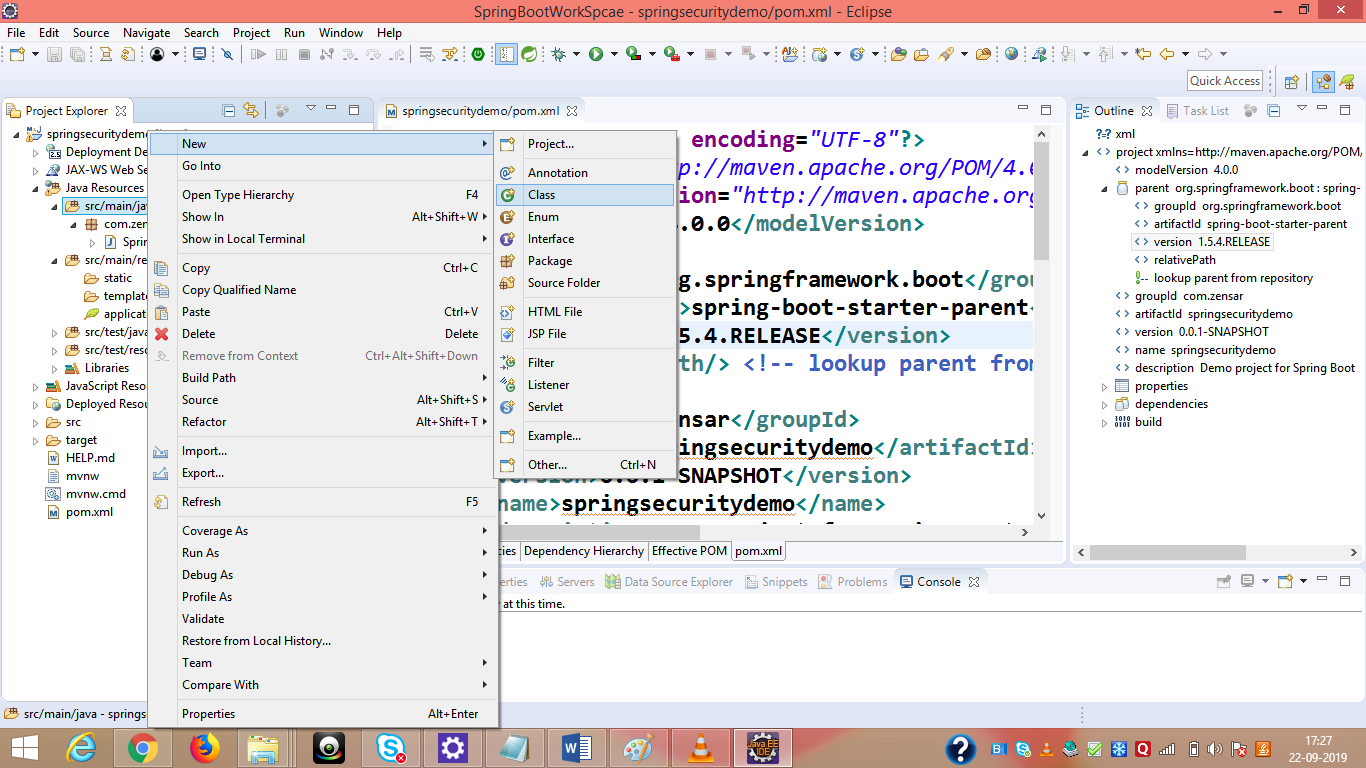


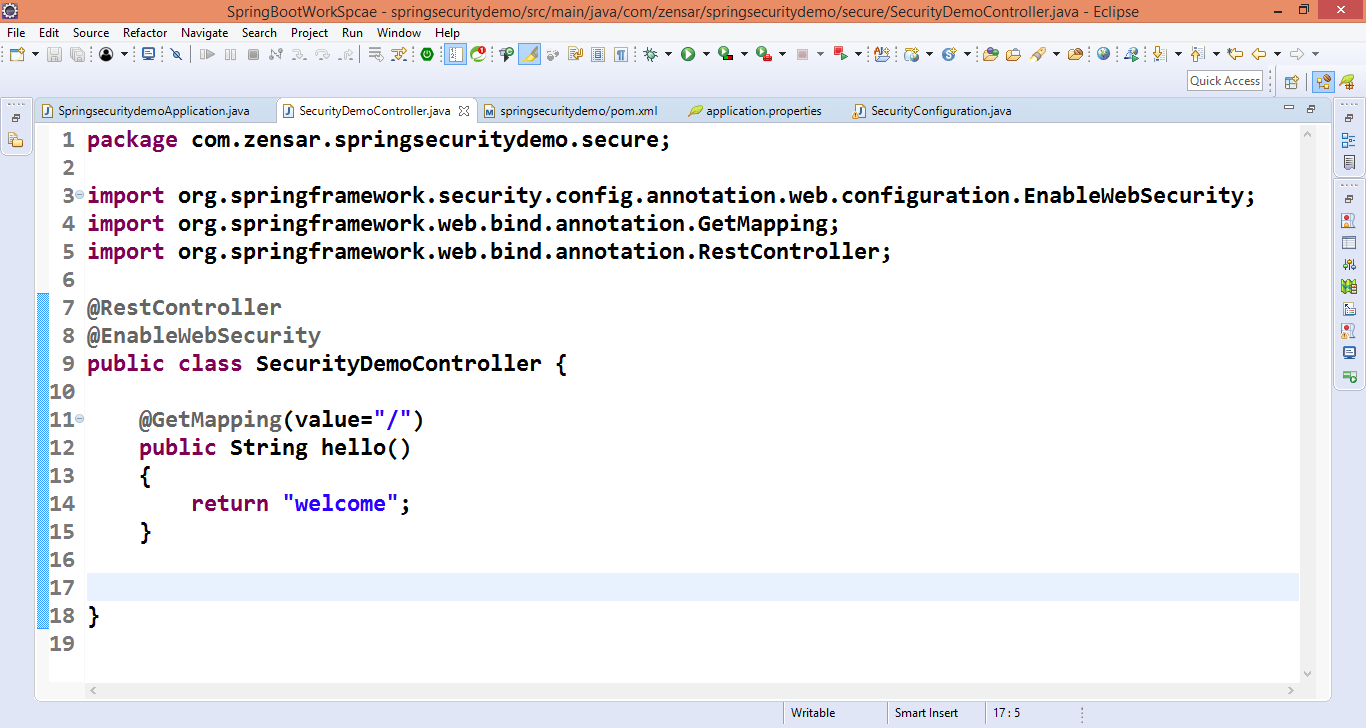


Choose boot version is 2.0.2 because older will not support to spring security

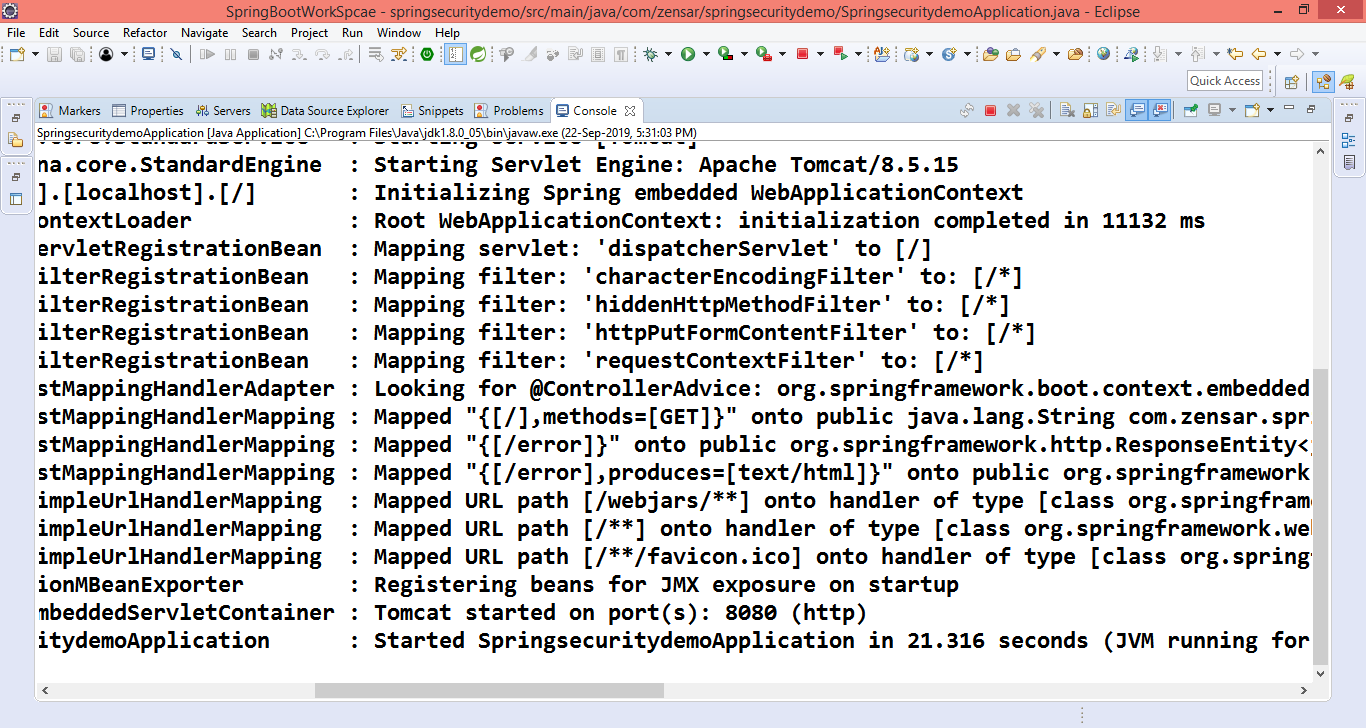


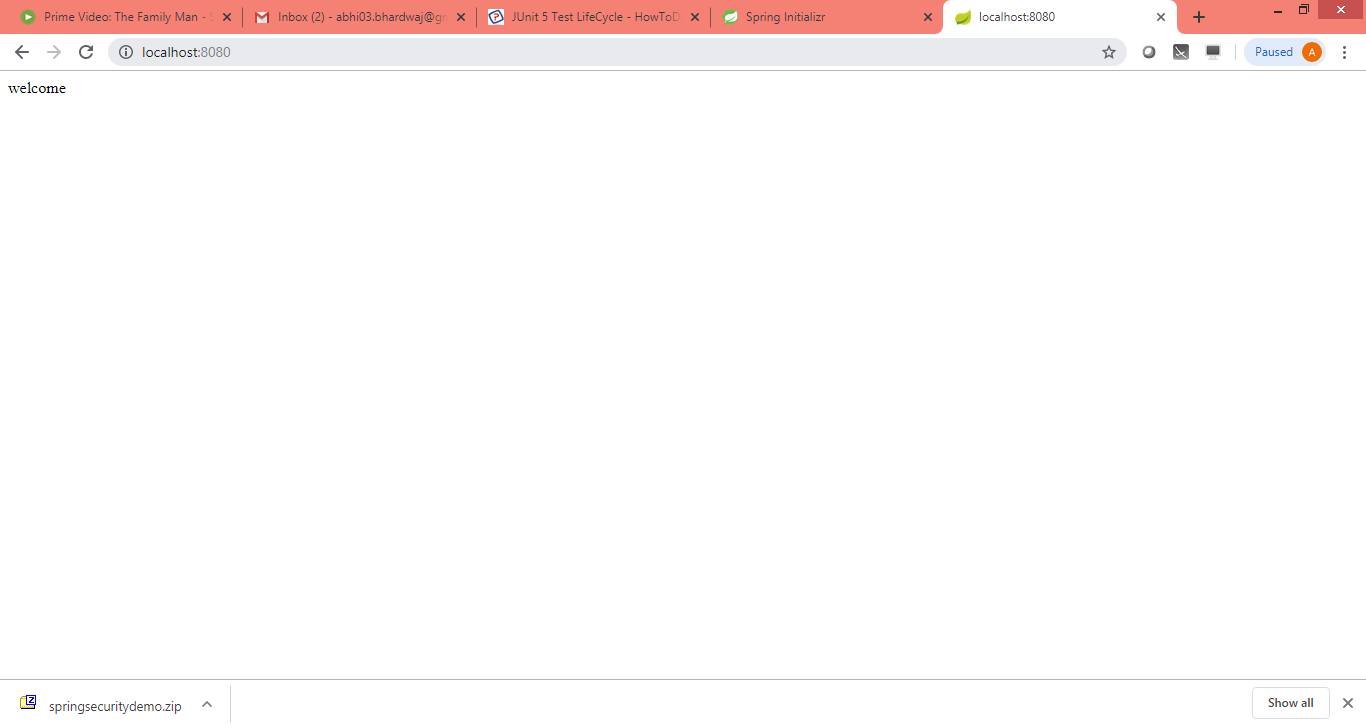
Create a controller class



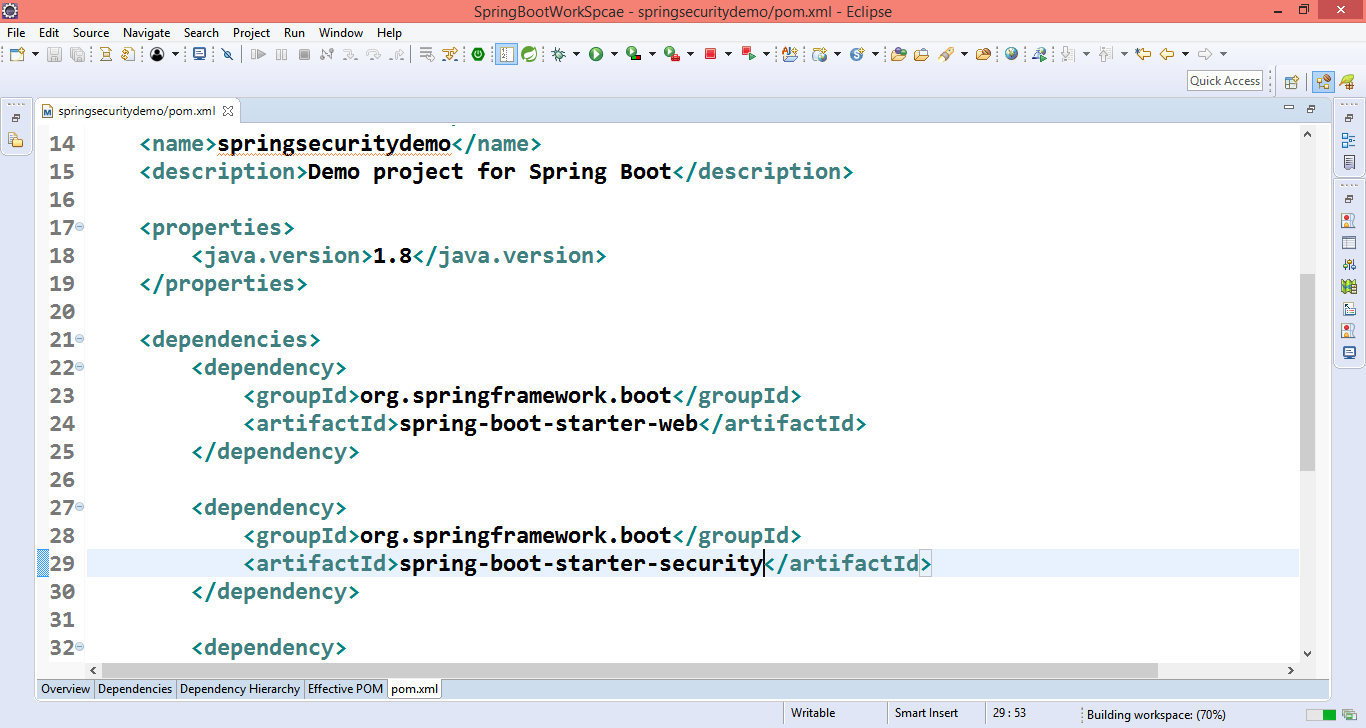


Run this project

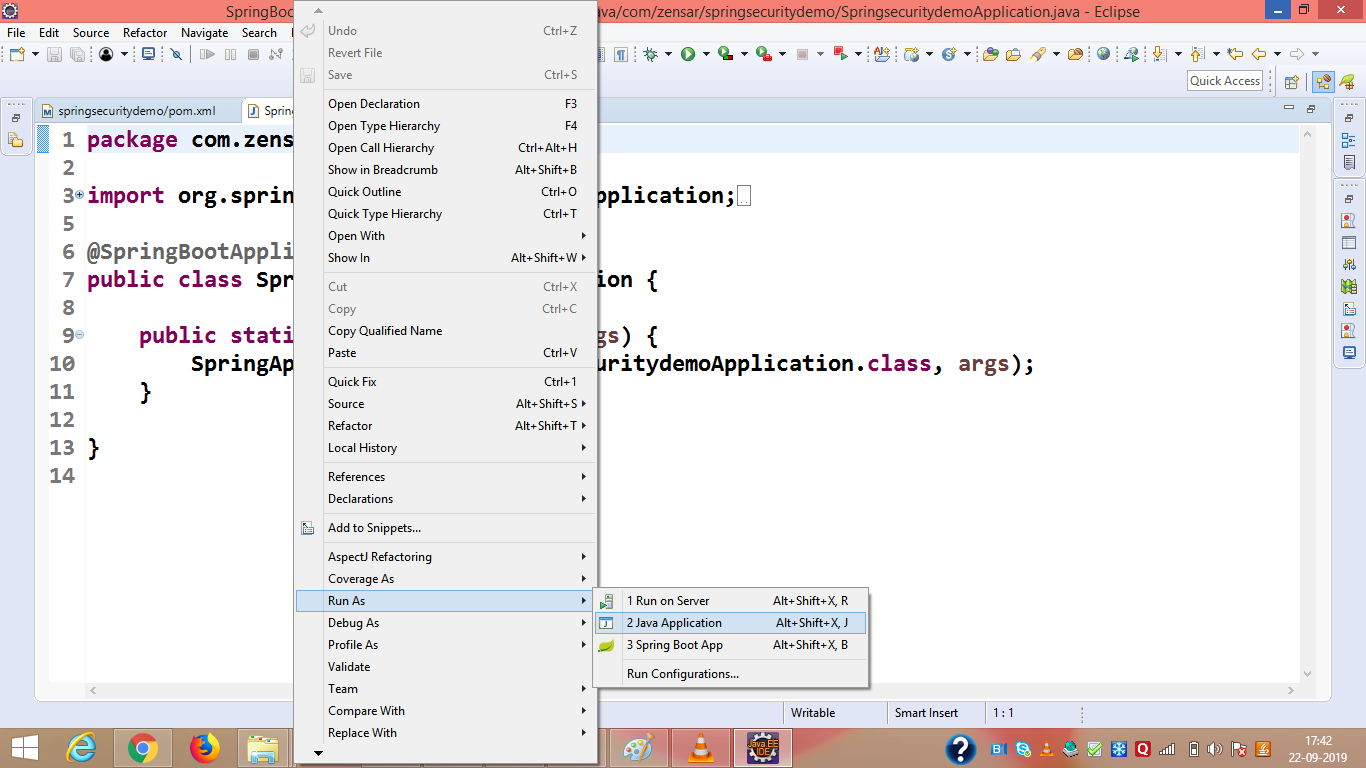


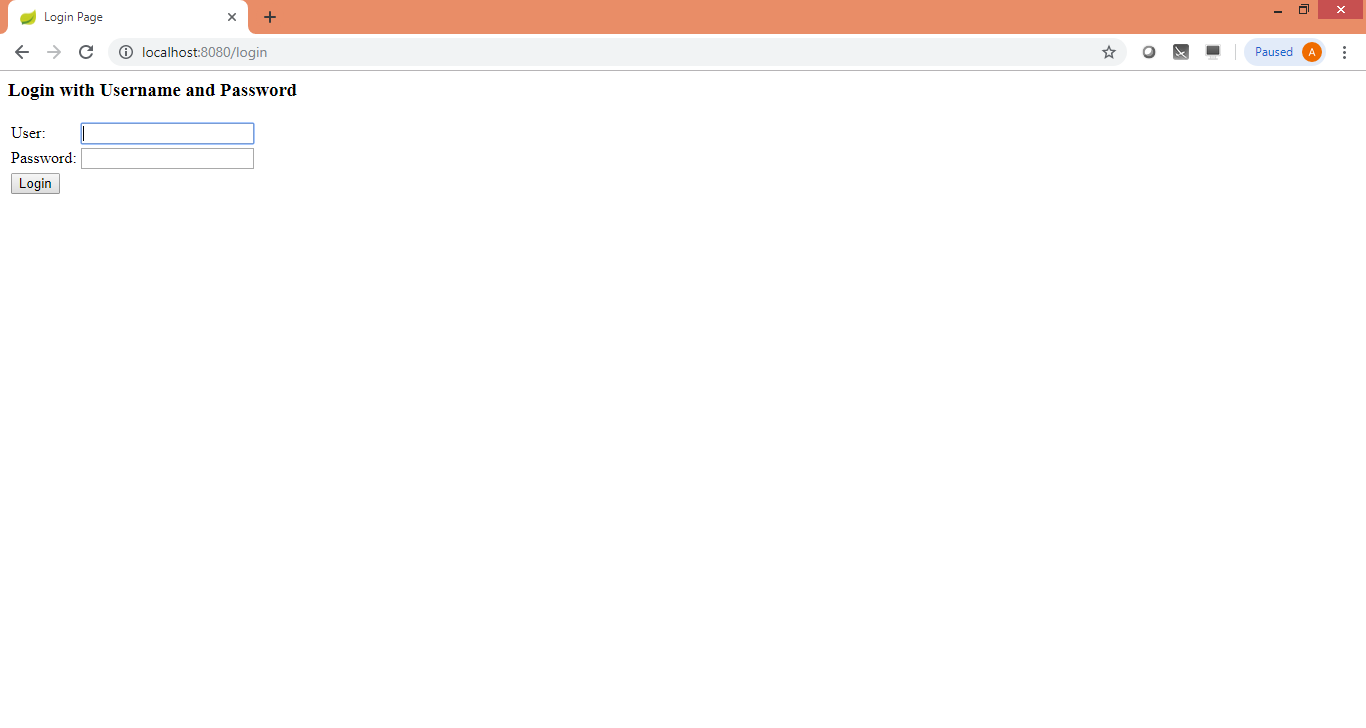


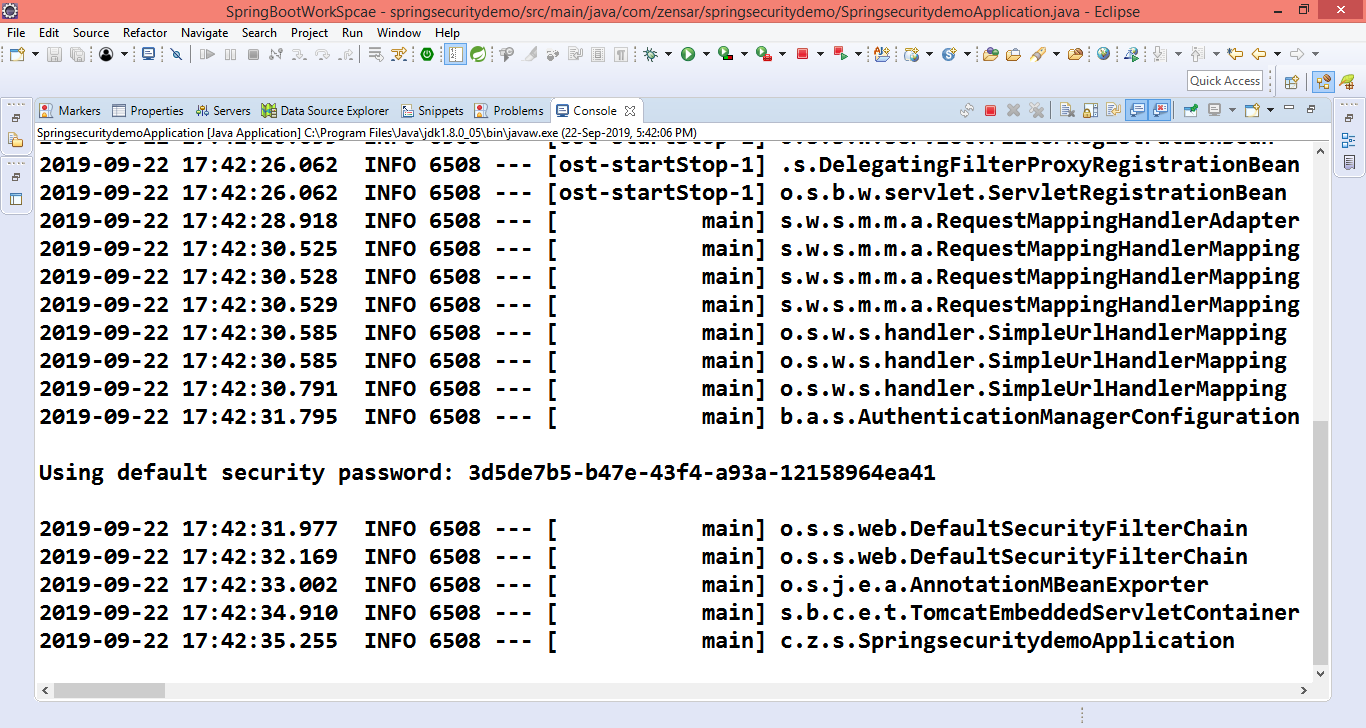
**Add spring boot starter security in pom.xml file**



Run your project again



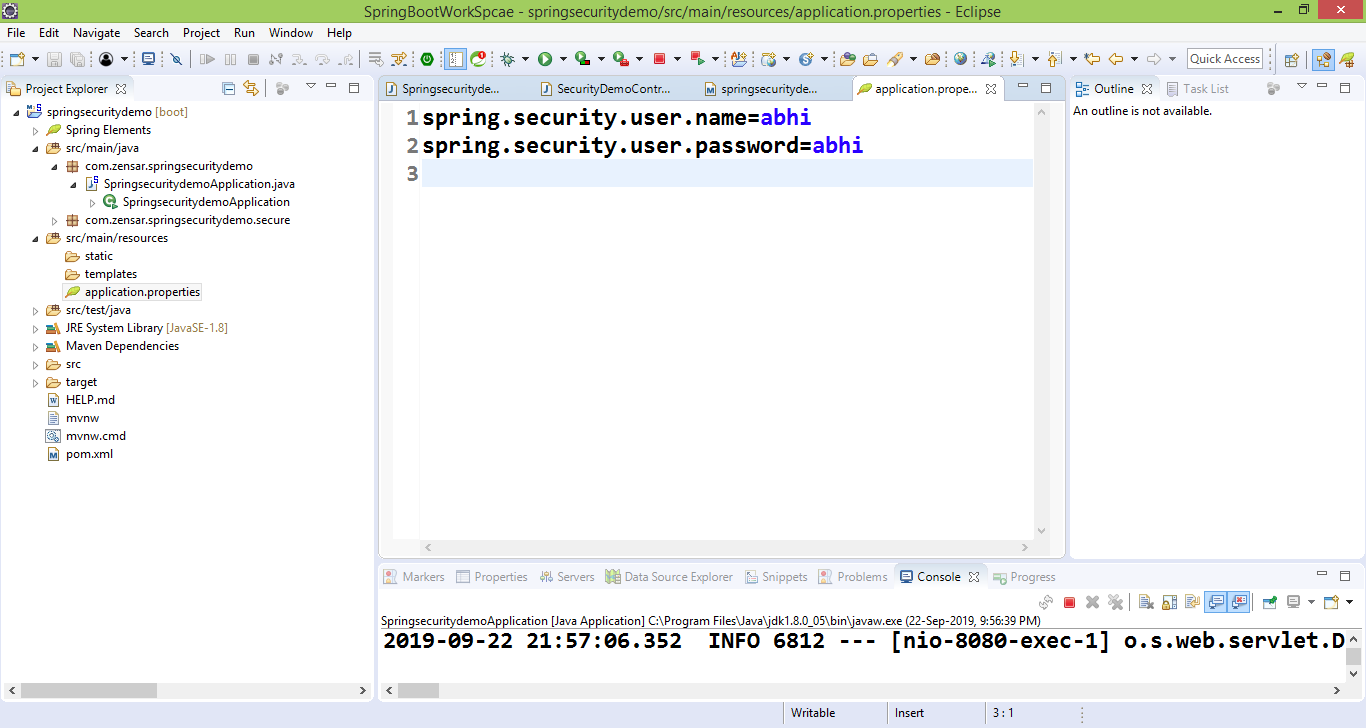




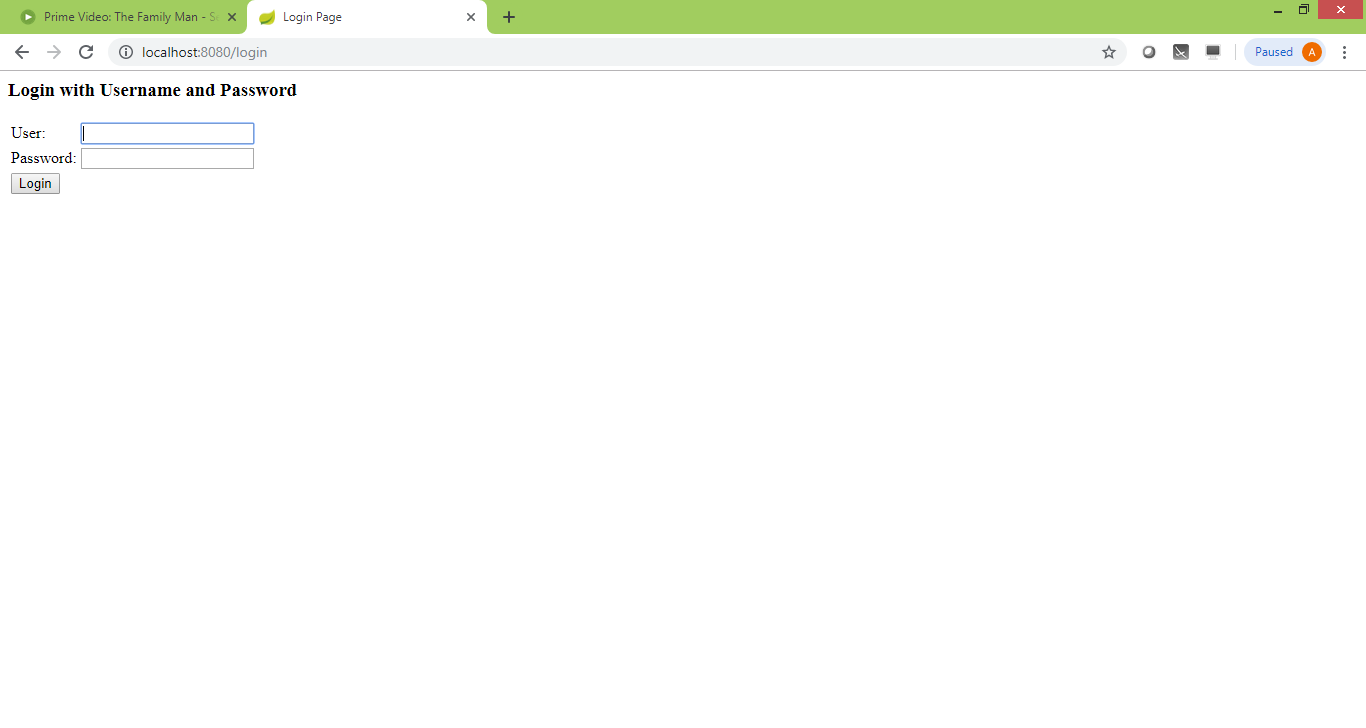
Note: When we add spring security it create default login page, with help of filter. Default user name is user and passowrd we can get from console, everytime it will generate separate password.

**If you want to your created user name and password**

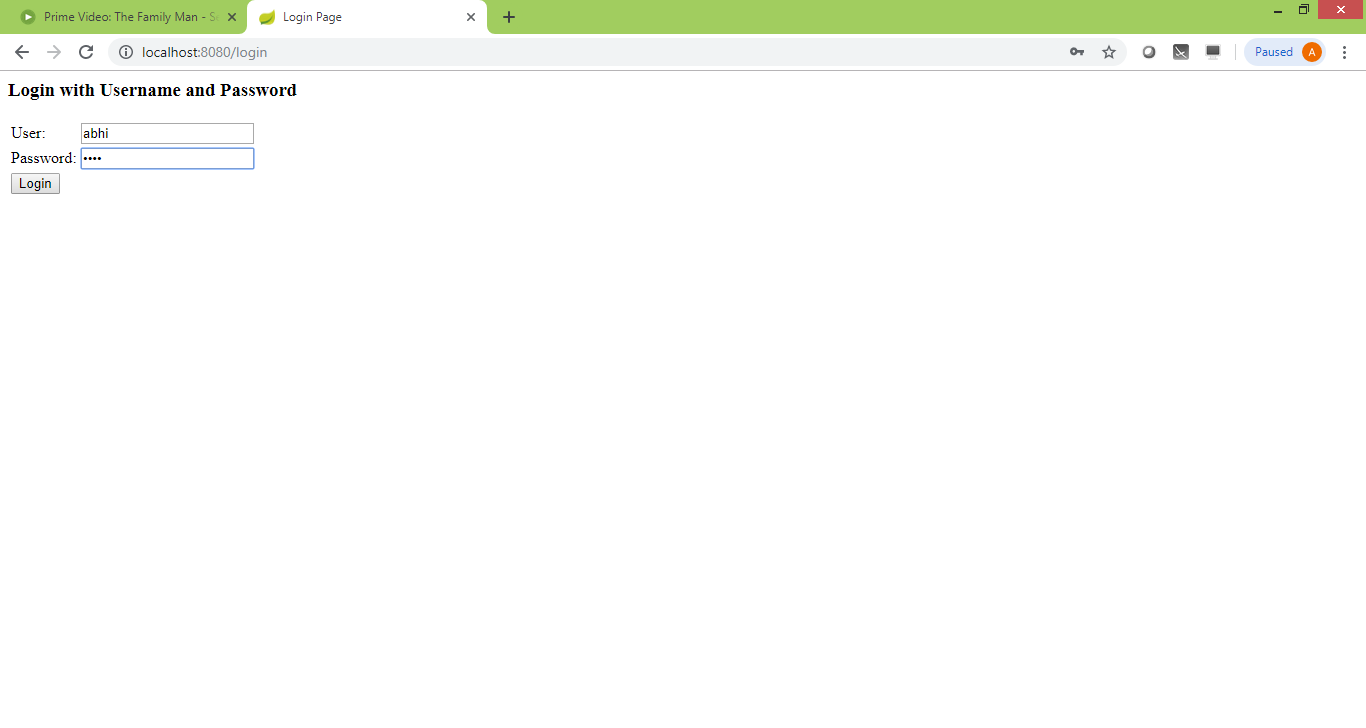
Goto application.properties file

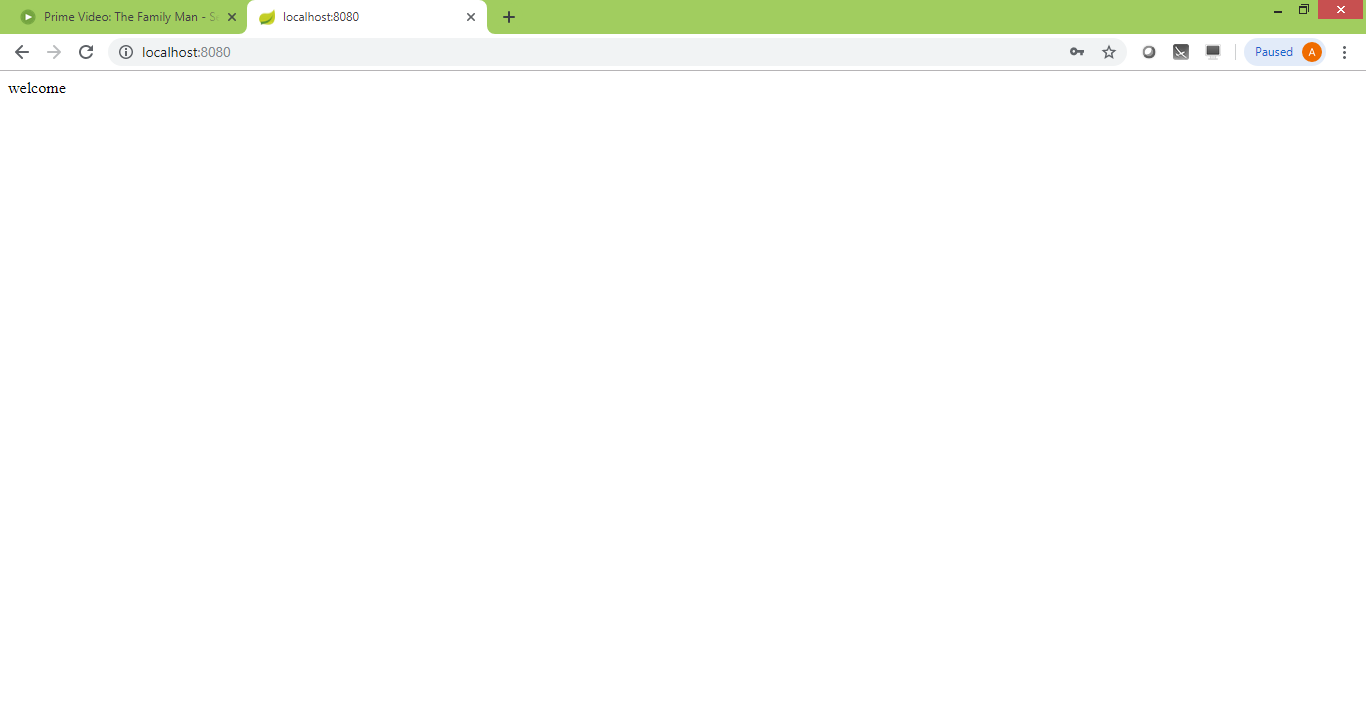


Now run again,it will not generate default credentials and take our defined in properties file.

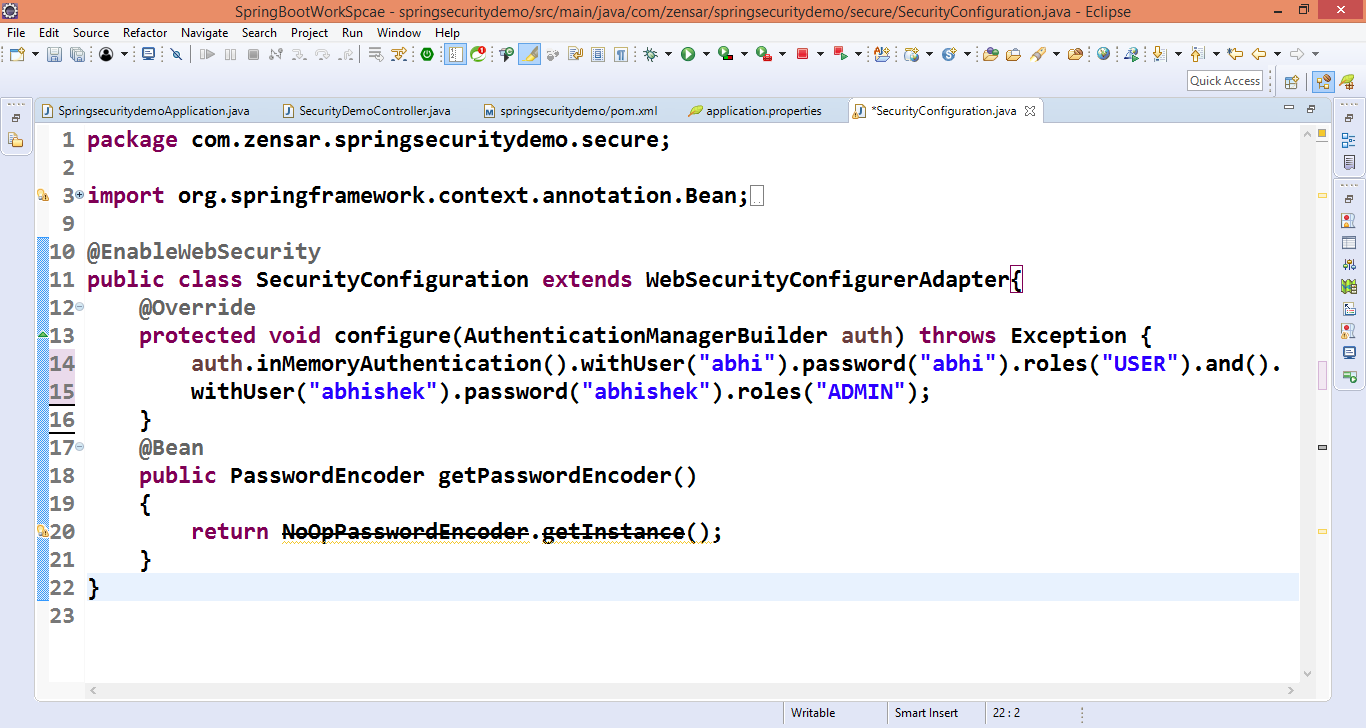


Put user name and password

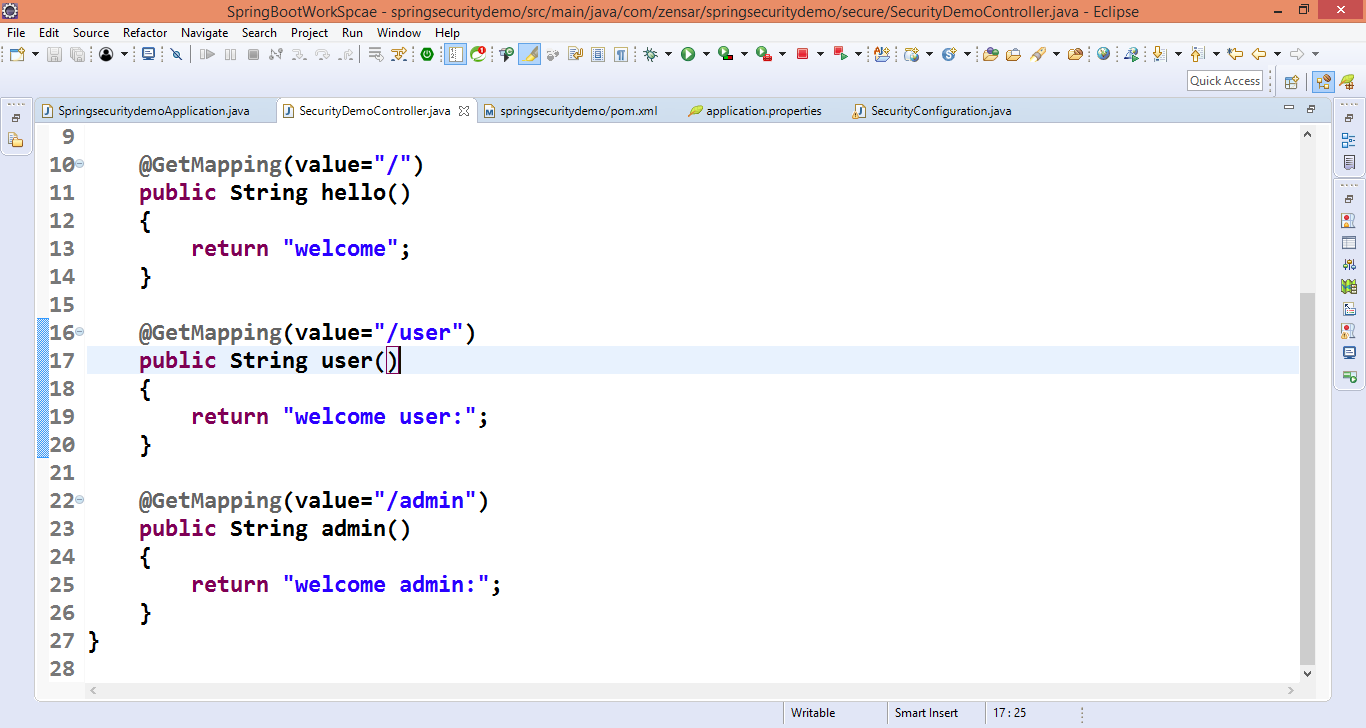


Create a new class SecurityConfiguration

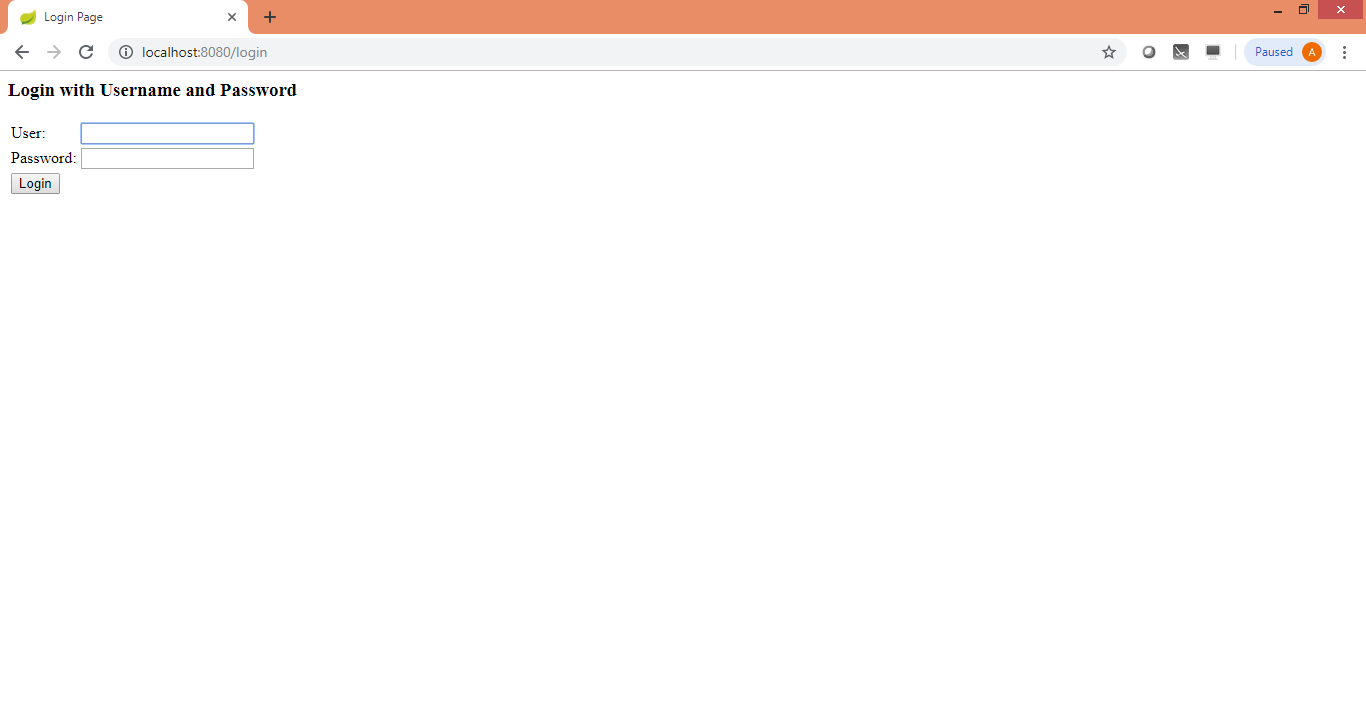
We create two users USER and ADMIN

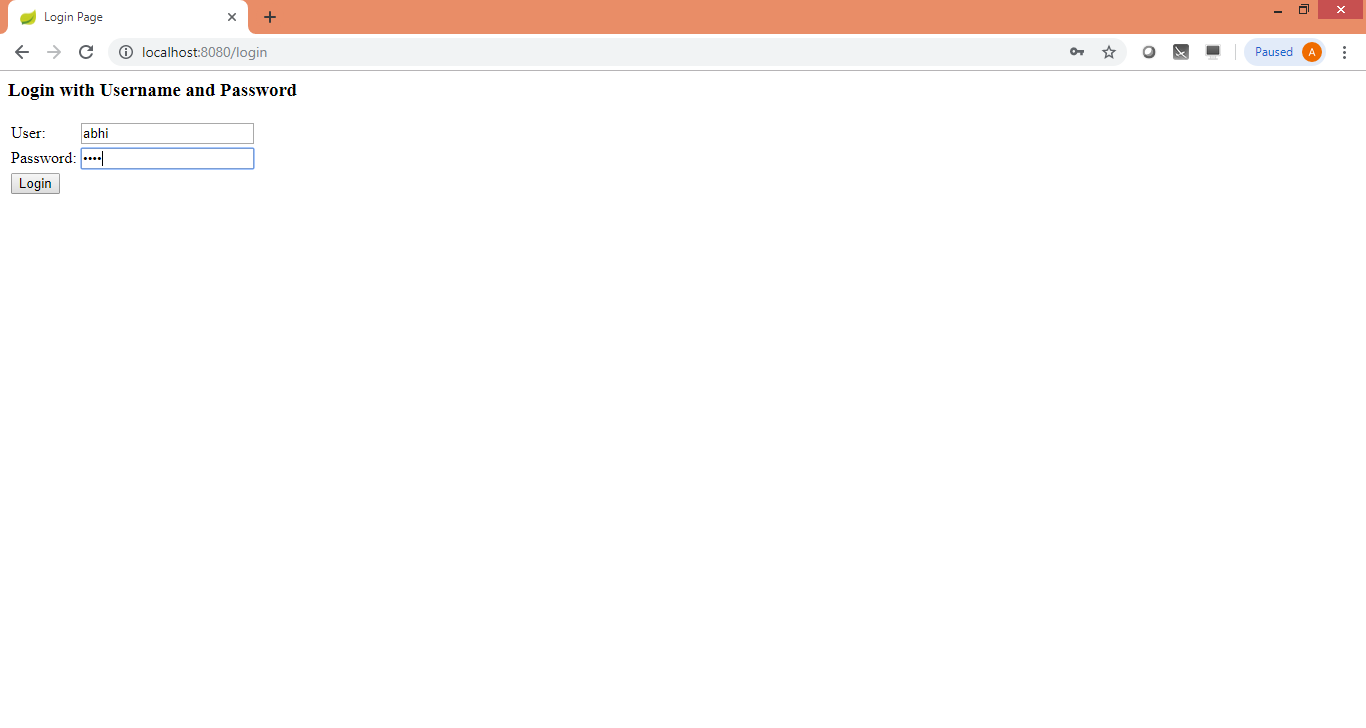


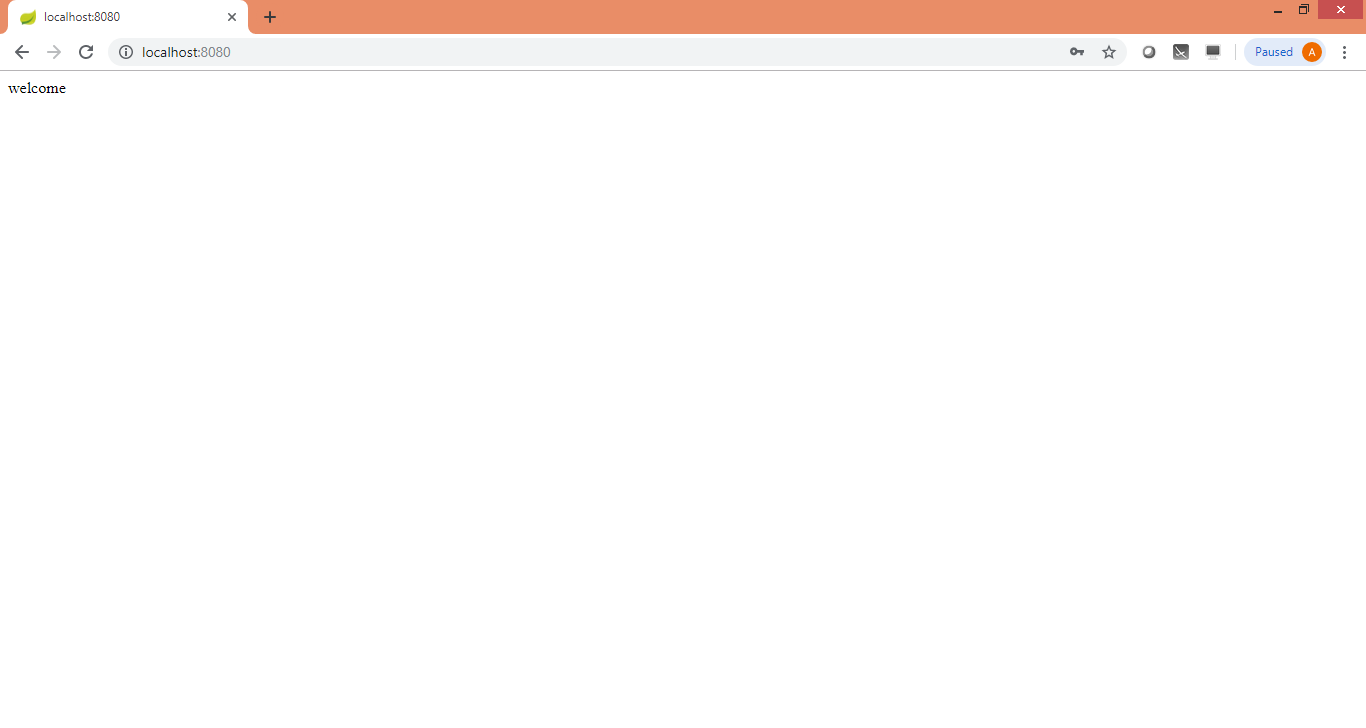
Create two URL in Controller class



Goto URL http://localhost:8080

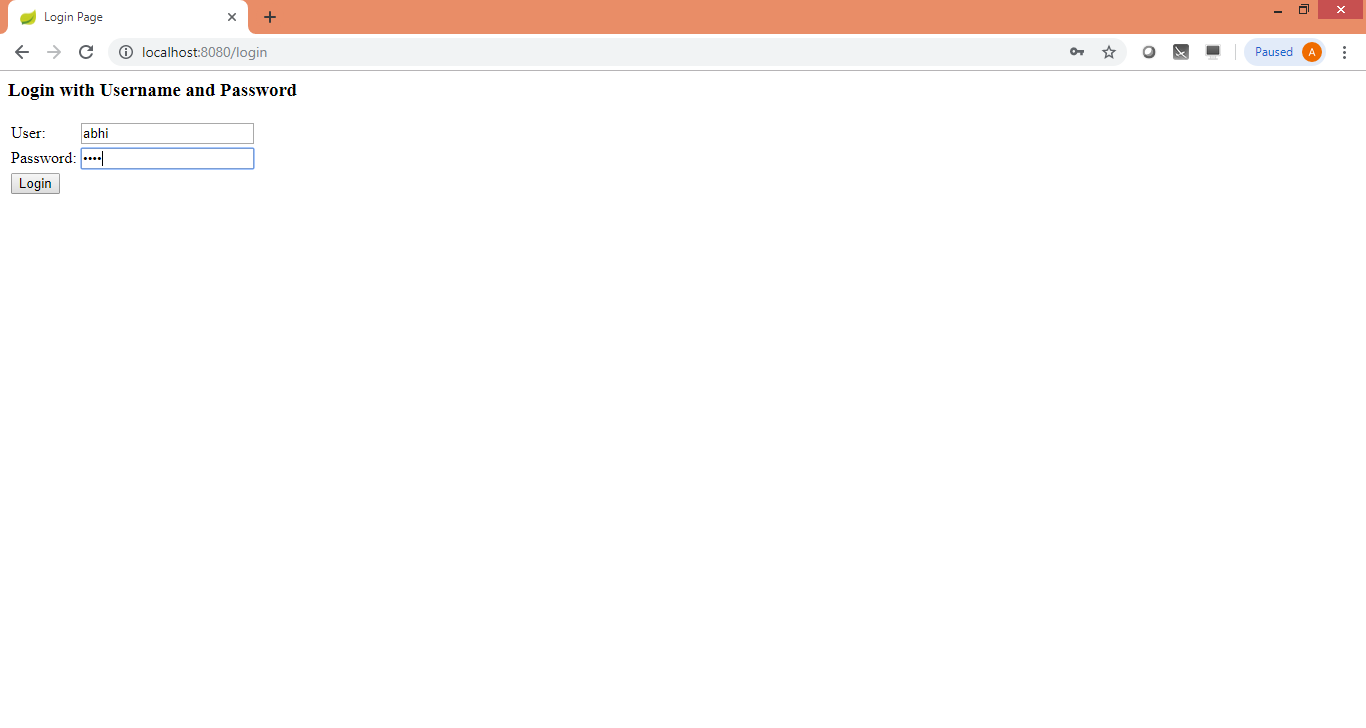


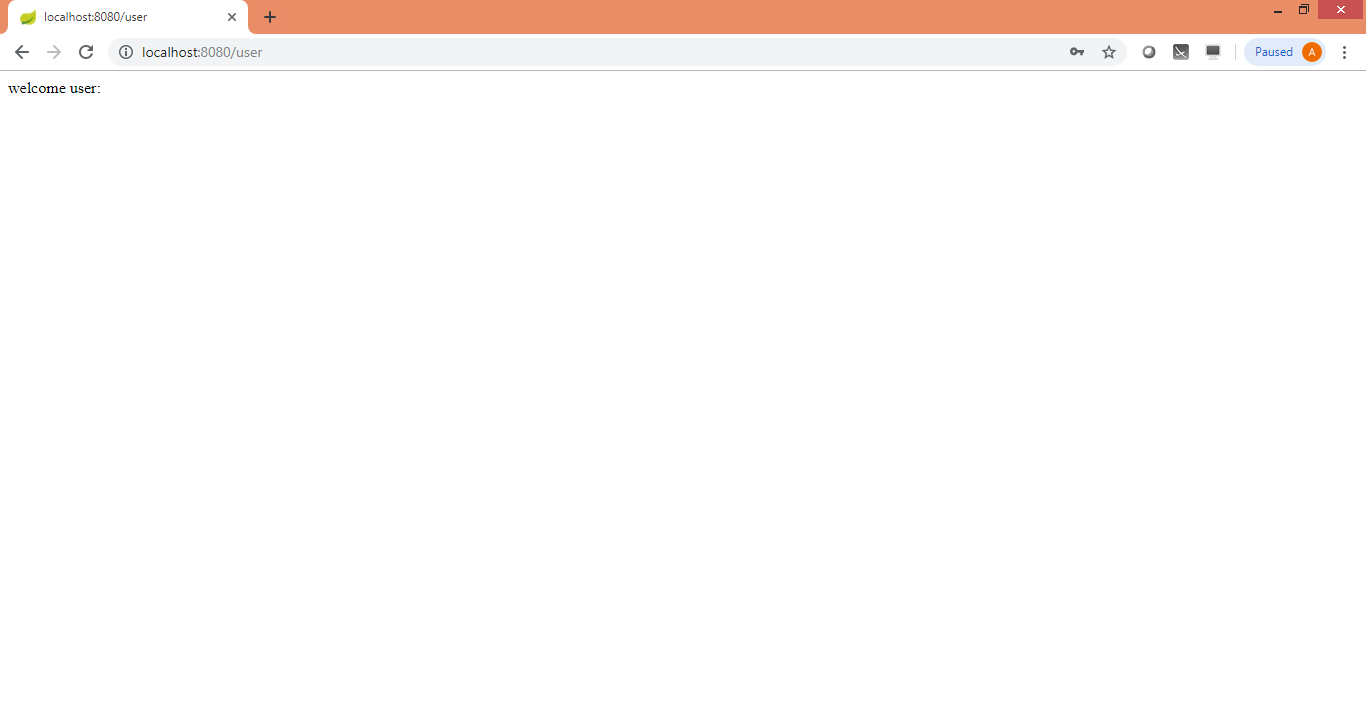




Now check another one

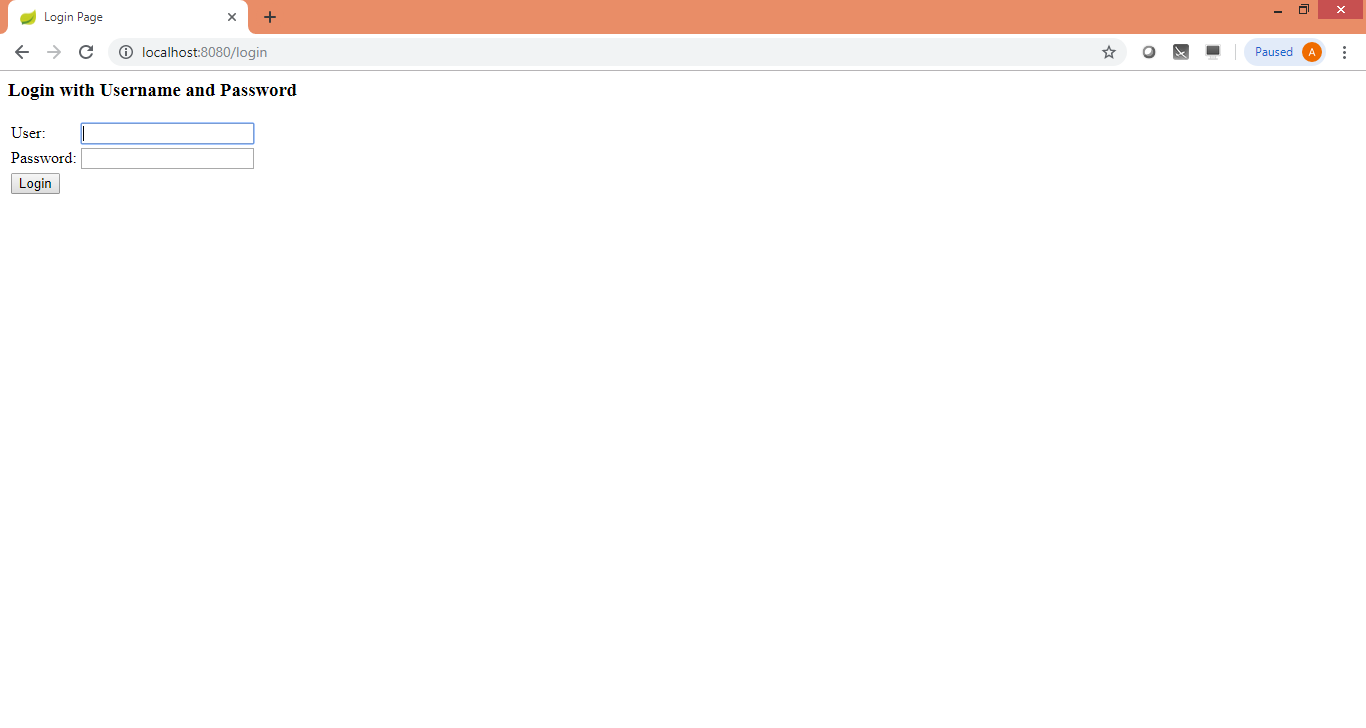
http://localhost:8080/user

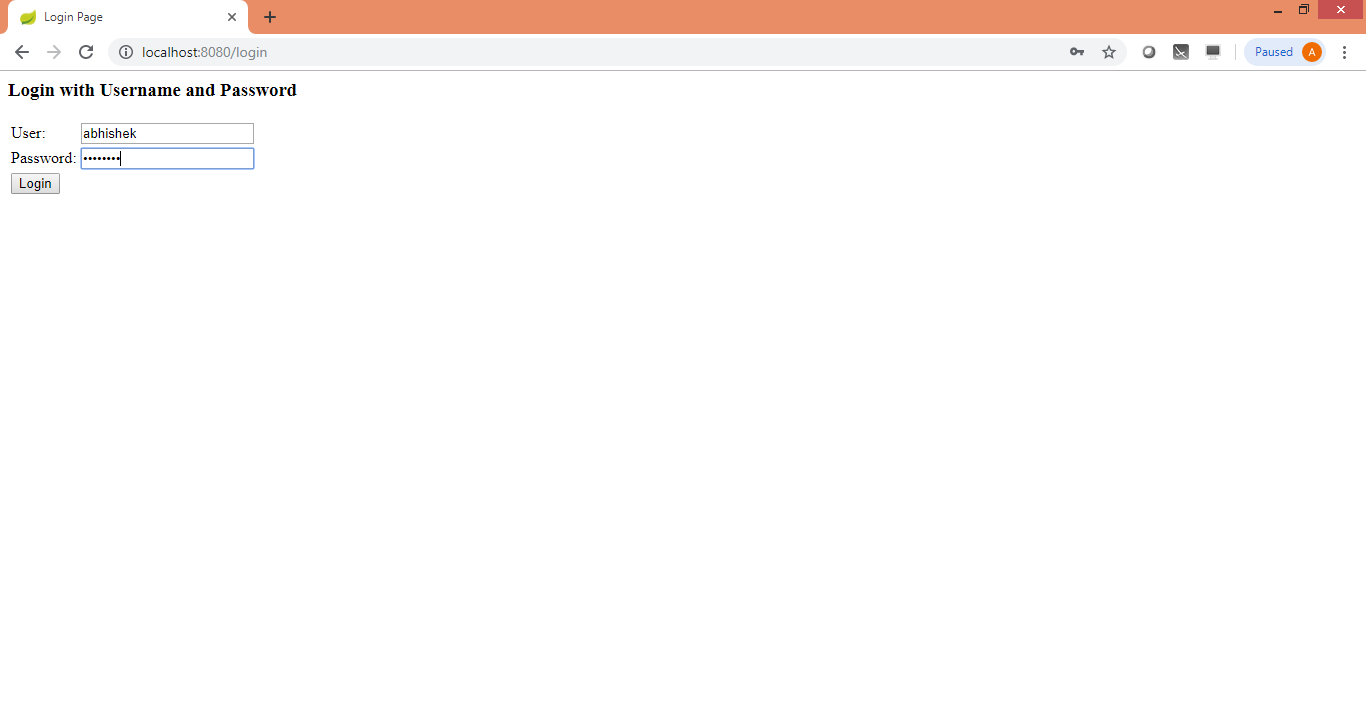


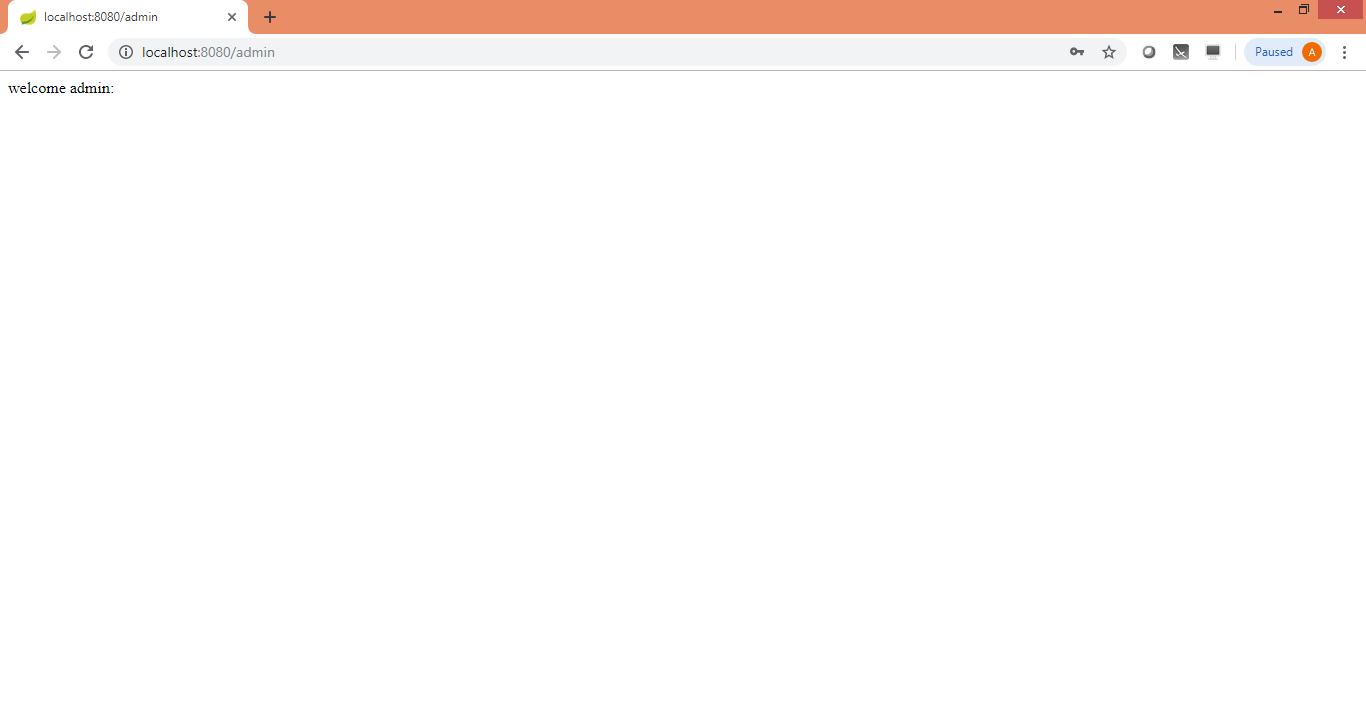


Check third one

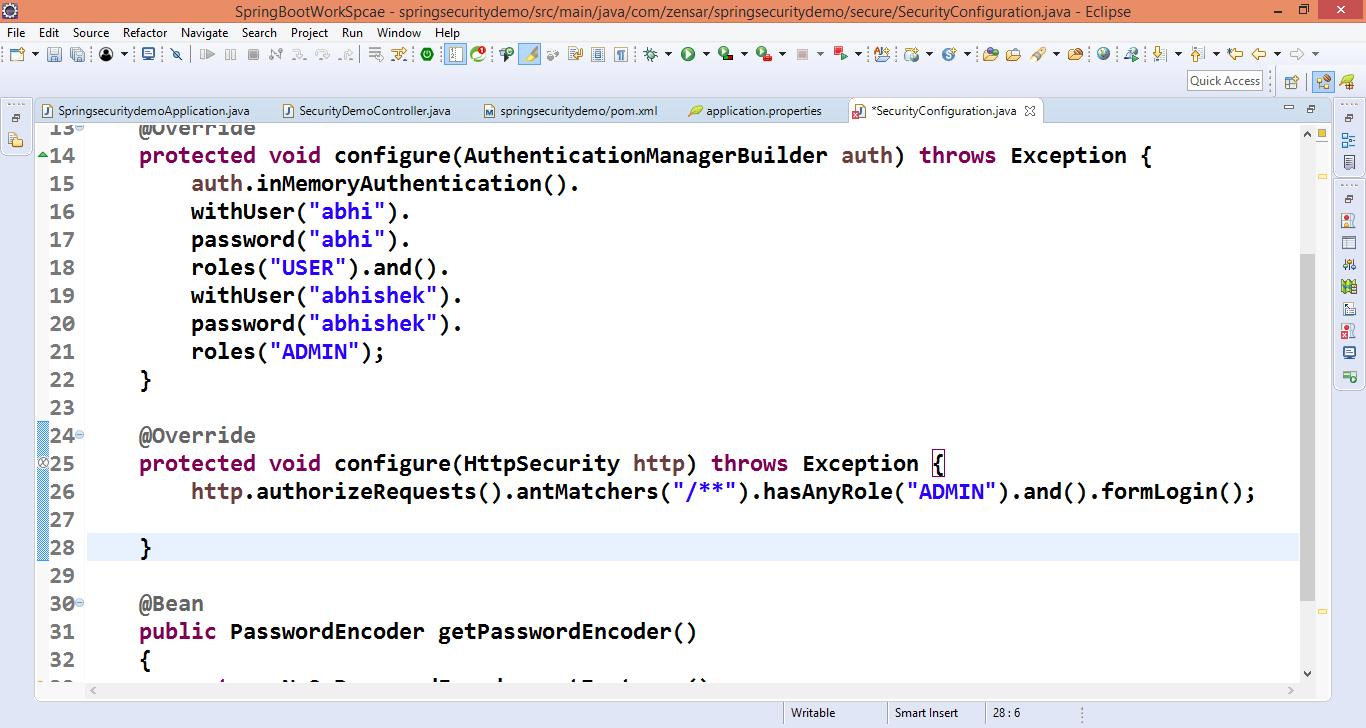
http://localhost:8080/admin







If you want to restrict specific for role we can do this



If you want to allow for every user

