Configuring Basic Security

Development Process

Step-By-Step

Development Process

Step-By-Step

1. Create Spring Security Initializer

Development Process

Step-By-Step

1. Create Spring Security Initializer

2. Create Spring Security Configuration (@Configuration)

Development Process

Step-By-Step

1. Create Spring Security Initializer

2. Create Spring Security Configuration (@Configuration)

3. Add users, passwords and roles

Spring Security Web App Initializer

Spring Security Web App Initializer

- Spring Security provides support for security initialization

Spring Security Web App Initializer

- Spring Security provides support for security initialization
- Your security code is used to initialize the servlet container

Spring Security Web App Initializer

- Spring Security provides support for security initialization
- Your security code is used to initialize the servlet container
- Special class to register the **Spring Security Filters**

Spring Security Web App Initializer

- Spring Security provides support for security initialization
- Your security code is used to initialize the servlet container
- Special class to register the **Spring Security Filters**

AbstractSecurityWebApplicationInitializer

Spring Security Web App Initializer

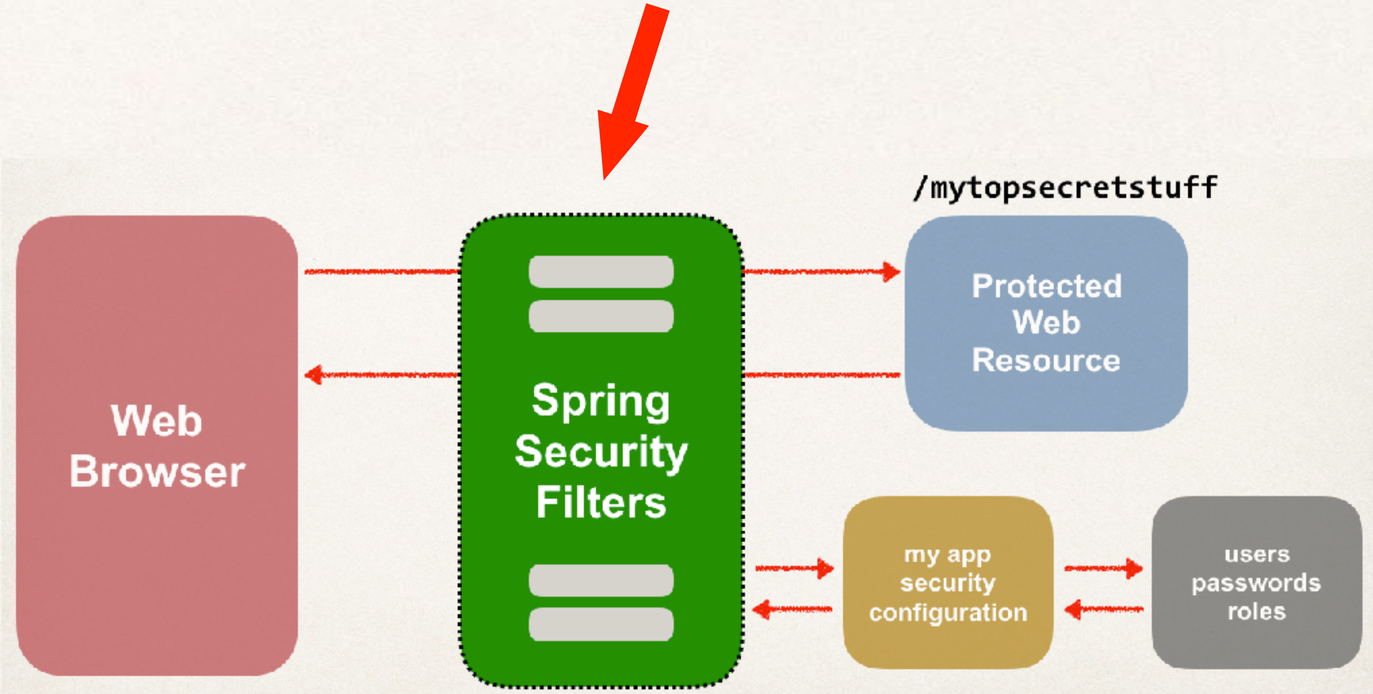
Spring Security Web App Initializer

AbstractSecurityWebApplicationInitializer

Spring Security Web App Initializer

AbstractSecurityWebApplicationInitializer

- Special class to register the **Spring Security Filters**



Spring Security Web App Initializer (more info)

AbstractSecurityWebApplicationInitializer

Spring Security Web App Initializer (more info)

AbstractSecurityWebApplicationInitializer

- Your TO DO list

Spring Security Web App Initializer (more info)

AbstractSecurityWebApplicationInitializer

- Your TO DO list
  - Extend this abstract base class

Spring Security Web App Initializer (more info)

AbstractSecurityWebApplicationInitializer

- Your TO DO list
  - Extend this abstract base class
  - that's it!

Step 1: Create Spring Security Initializer

File: SecurityWebApplicationInitializer.java

```
import org.springframework.security.web.context.AbstractSecurityWebApplicationInitializer;

public class SecurityWebApplicationInitializer extends AbstractSecurityWebApplicationInitializer {

}
```

Step 2: Create Spring Security Configuration

File: DemoSecurityConfig.java

```
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

@Configuration
@EnableWebSecurity
public class DemoSecurityConfig extends WebSecurityConfigurerAdapter {

}
```

Step 3: Add users, passwords and roles

File: DemoSecurityConfig.java

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

  @Override
  protected void configure(AuthenticationManagerBuilder auth) throws Exception {

    // add our users for in memory authentication









  }
```

Step 3: Add users, passwords and roles

File: DemoSecurityConfig.java

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

  @Override
  protected void configure(AuthenticationManagerBuilder auth) throws Exception {

    // add our users for in memory authentication

    UserBuilder users = User.withDefaultPasswordEncoder();

    auth.inMemoryAuthentication()
      .withUser(users.username("john").password("test123").roles("EMPLOYEE"))
      .withUser(users.username("mary").password("test123").roles("MANAGER"))
      .withUser(users.username("susan").password("test123").roles("ADMIN"));

  }
```

Step 3: Add users, passwords and roles

File: DemoSecurityConfig.java

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

  @Override
  protected void configure(AuthenticationManagerBuilder auth) throws Exception {

    // add our users for in memory authentication

    UserBuilder users = User.withDefaultPasswordEncoder();

    auth.inMemoryAuthentication()
      .withUser(users.username("john").password("test123").roles("EMPLOYEE"))
      .withUser(users.username("mary").password("test123").roles("
      .withUser(users.username("susan").password("test123").roles(

  }
```

We will add DB support in later videos

(plaintext and encrypted)

Step 3: Add users, passwords and roles

File: DemoSecurityConfig.java

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

  @Override
  protected void configure(AuthenticationManagerBuilder auth) throws Exception {

    // add our users for in memory authentication

    UserBuilder users = User.withDefaultPasswordEncoder();

    auth.inMemoryAuthentication()
      .withUser(users.username("john").password("test123").roles("EMPLOYEE"))
      .withUser(users.username("mary").password("test123").roles("MANAGER"))
      .withUser(users.username("susan").password("test123").roles("ADMIN"));

  }
```