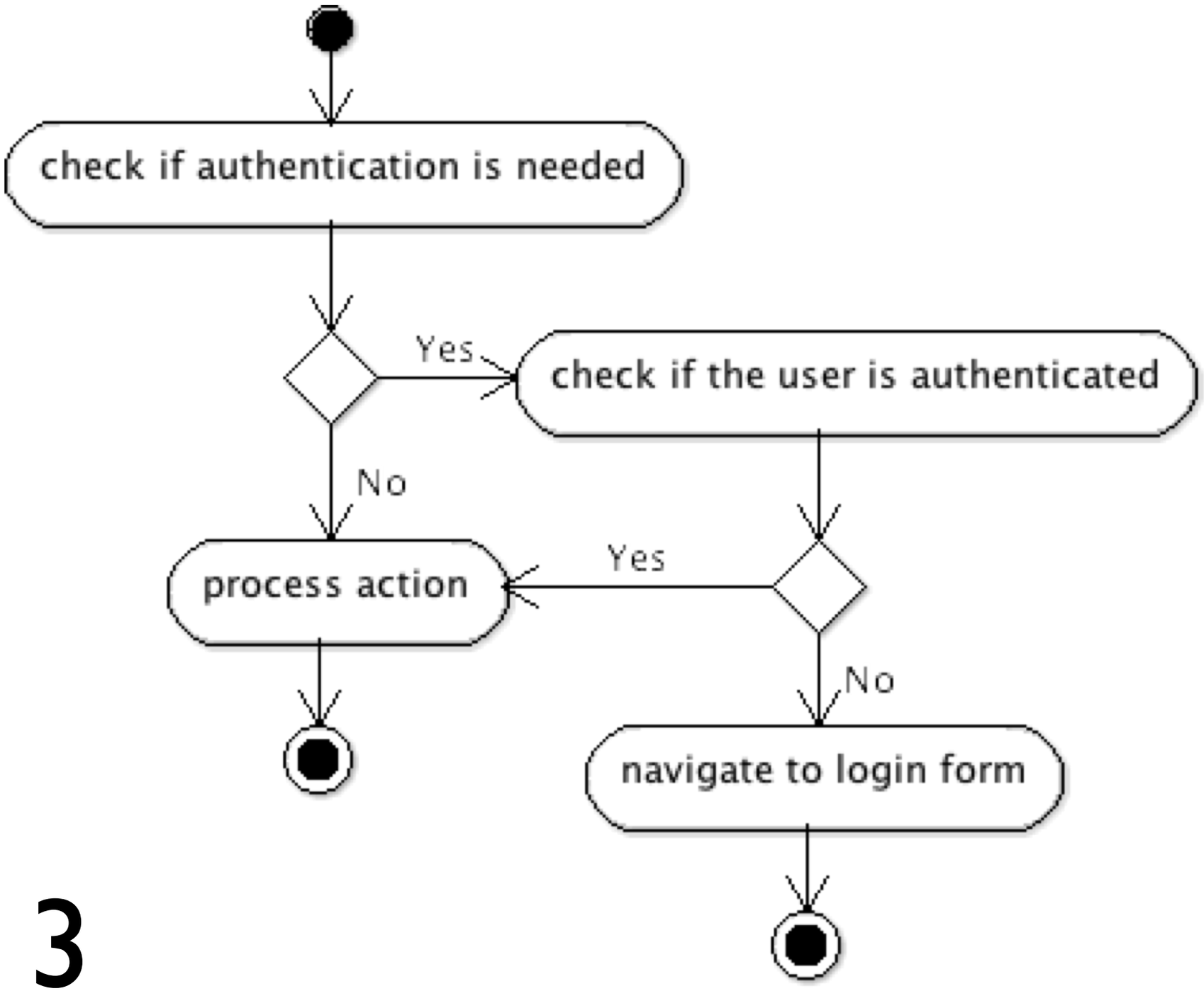
Security

Spring

STEP 1

USER CLICKS ON A LINK OR BUTTON…



CONTROLLER

```
private void
    processRequest(request, response){
    // check if authentication is needed
    // check if user is authenticated
    // if not authenticated and authentication needed
     => navigate to login form
    // else continue and perform action
}
```

see next pages

// check authentication needed

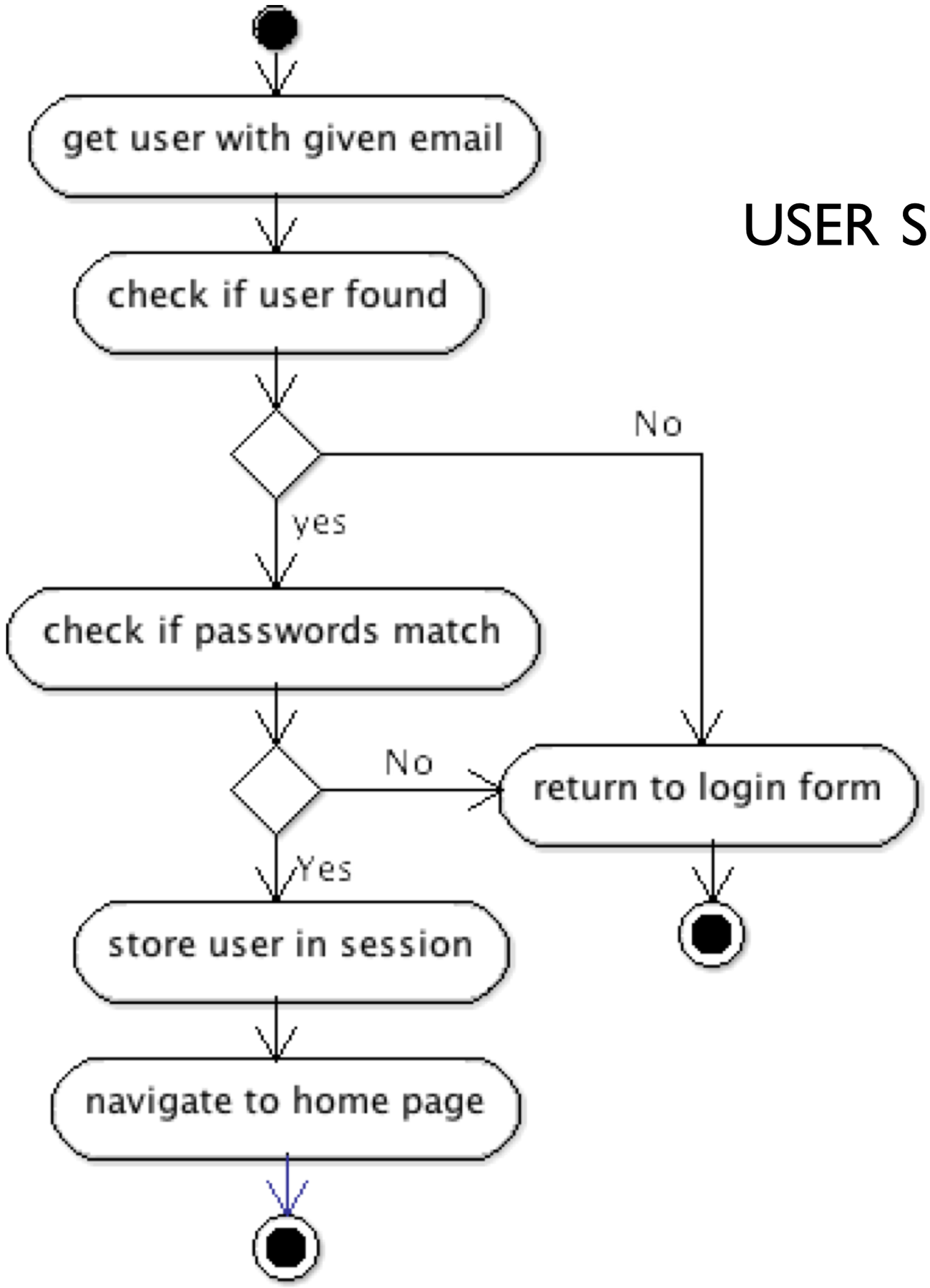
```
    private boolean
        isAuthenticationNeeded(action){
      // if action = ….return true
      // if action = … return false
}
```

WEB 3

// check if user is authenticated

because we need
session from request

```
private boolean
    isAuthenticated(request){
    // get session from request
    // get person-object from session
    // if no person found => return false
    // else return true
}
```



STEP 2

USER SUBMITS LOGIN FORM…

LOGIN: VIEW

- login.jsp:
  - username
  - password
  - action= “login”

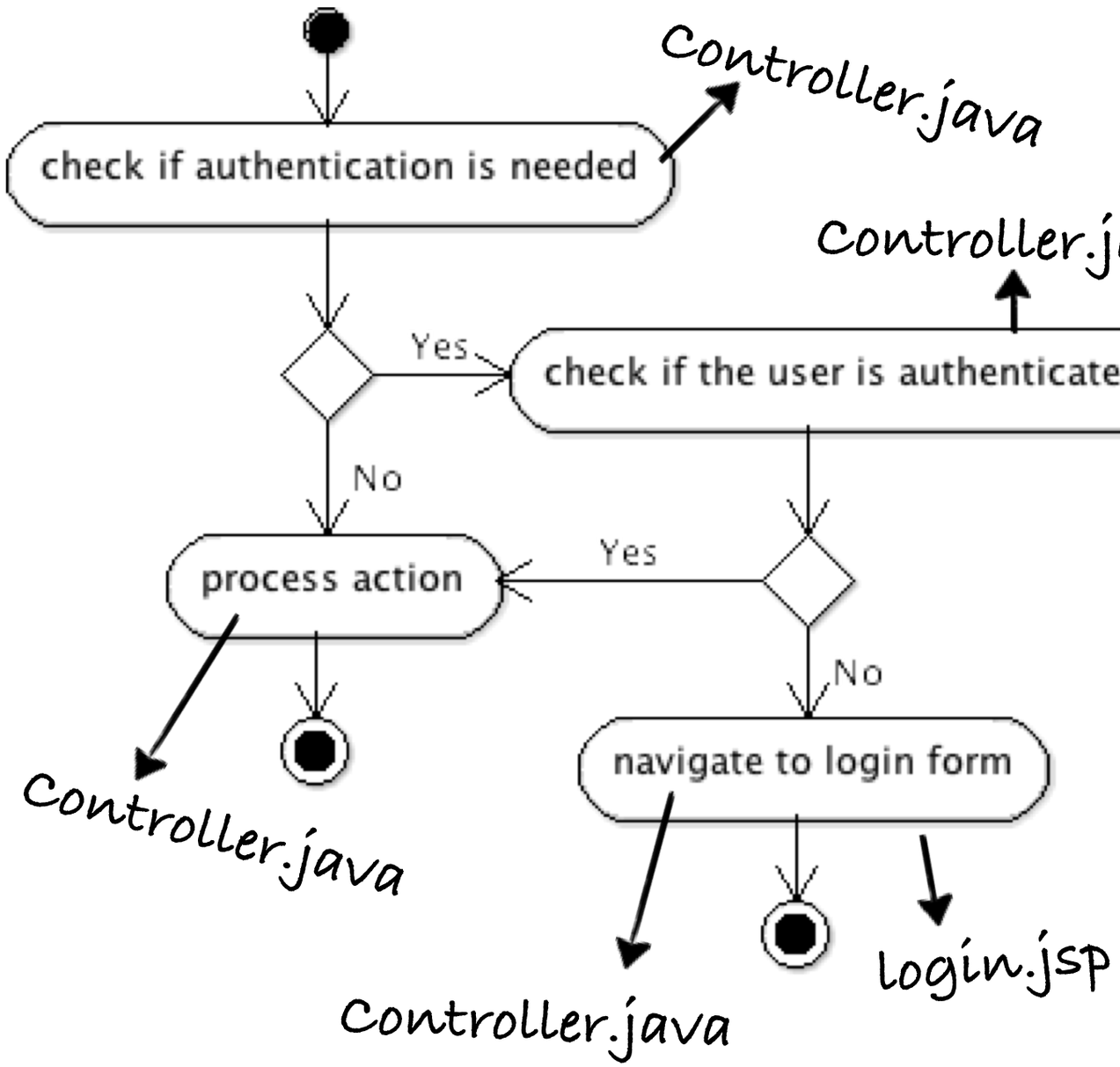
LOGIN: CONTROLLER

```
private void login(request, response){

    // get user from service
    // if user found
        // create session
        // store person-object in session
        // continue with action
    // else
        // create error message
        // return to login form
}
```

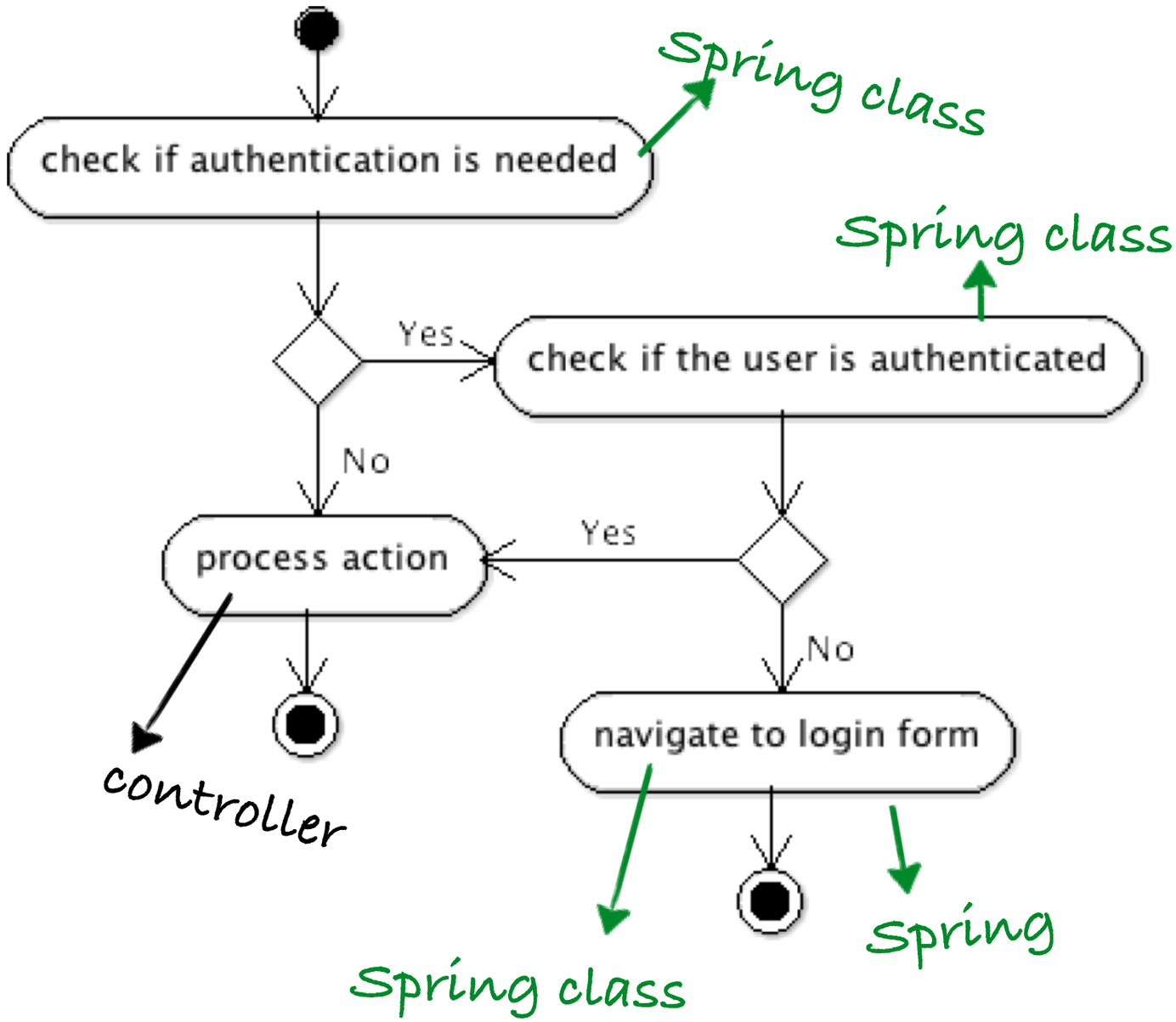
see next page

LOGIN: MODEL - SERVICE

```
public Person
    getAuthenticatedUser(email, password){
    //get user from db
    //check password
    //if OK return user
    //else return null
}
```


check if authentication is needed
Controller.java
Controller.java
Yes
check if the user is authenticated
No
process action
Yes
No
navigate to login form
Controller.java
Controller.java
login.jsp

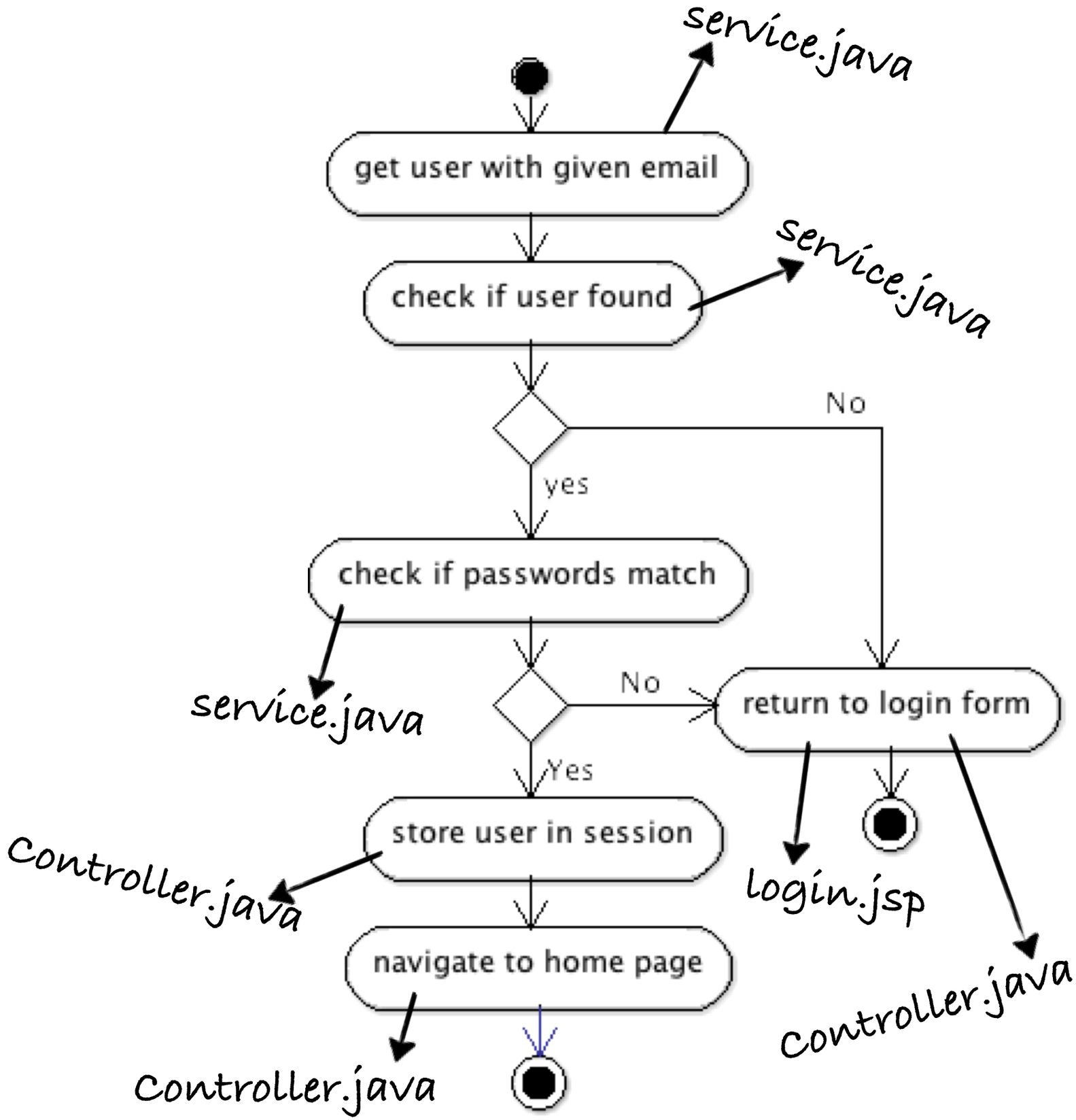
WEB 3

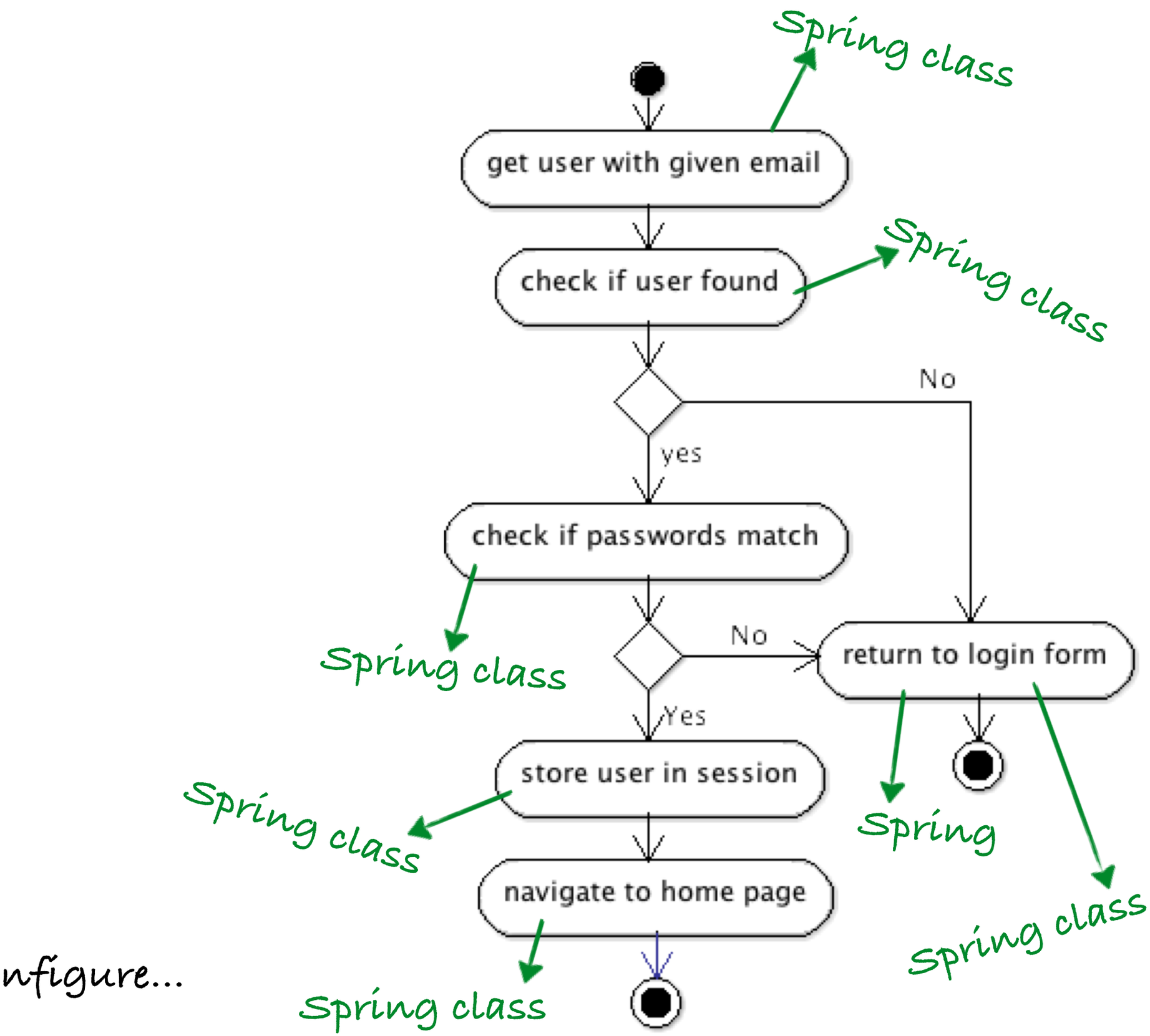


→ configure...

SPRING

WEB 3


service.java
get user with given email
check if user found
service.java
No
yes
check if passwords match
service.java
No
return to login form
Yes
store user in session
Controller.java
login.jsp
navigate to home page
Controller.java
Controller.java



configure...

SPRING

To do

Configure…

1. Spring filter to intercept the requests
2. who can do what
3. where to find the security info

DEPENDENCIES

pom.xml:

```
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-web</artifactId>
    <version>5.0.1.RELEASE</version>
</dependency>

<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-config</artifactId>
    <version>5.0.1.RELEASE</version>
</dependency>
```

add dependencies to pom.xml

1. Spring class
to intercept the requests

SecurityWebApplicationInitializer.java:

Create a NEW web initializer

```
public class SecurityWebApplicationInitializer extends
                        AbstractSecurityWebApplicationInitializer {

}
```

... to tell the container
to use this class as filter

2. Spring class to configure Security

WebSecurityConfig.java:

Create a NEW configuration file

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


}
```

... to configure authentication and authorization

2. **WHO** CAN DO WHAT

AUTHENTICATION

WebSecurityConfig.java:

… with a method to create a userservice with predifined users

```
@Bean
public UserDetailsService userDetailsService() {
    InMemoryUserDetailsManager manager =
                                        new InMemoryUserDetailsManager();
    manager.createUser(User.withDefaultPasswordEncoder()
            .username("miekem").password("1234")
            .roles("USER")
            .build());
    manager.createUser(User.withDefaultPasswordEncoder()
            .username("admin")
            .password("t")
            .roles("ADMIN").build());
    return manager;
}
```

2. Who can do **WHAT**

Authorization

WebSecurityConfig.java:

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http
        .authorizeRequests()
            .antMatchers("/", "/index.htm").permitAll()
            .antMatchers("/css/**", "/images/**").permitAll()
            .antMatchers("/country.htm").hasAnyRole("USER","ADMIN")
            .antMatchers("/country/**").hasRole("ADMIN")
            .anyRequest().authenticated()
            .and()
        .formLogin()
            .and()
        .httpBasic();
}
```

… and a method to configure authorization

3. WHERE TO FIND

THE SECURITY INFO

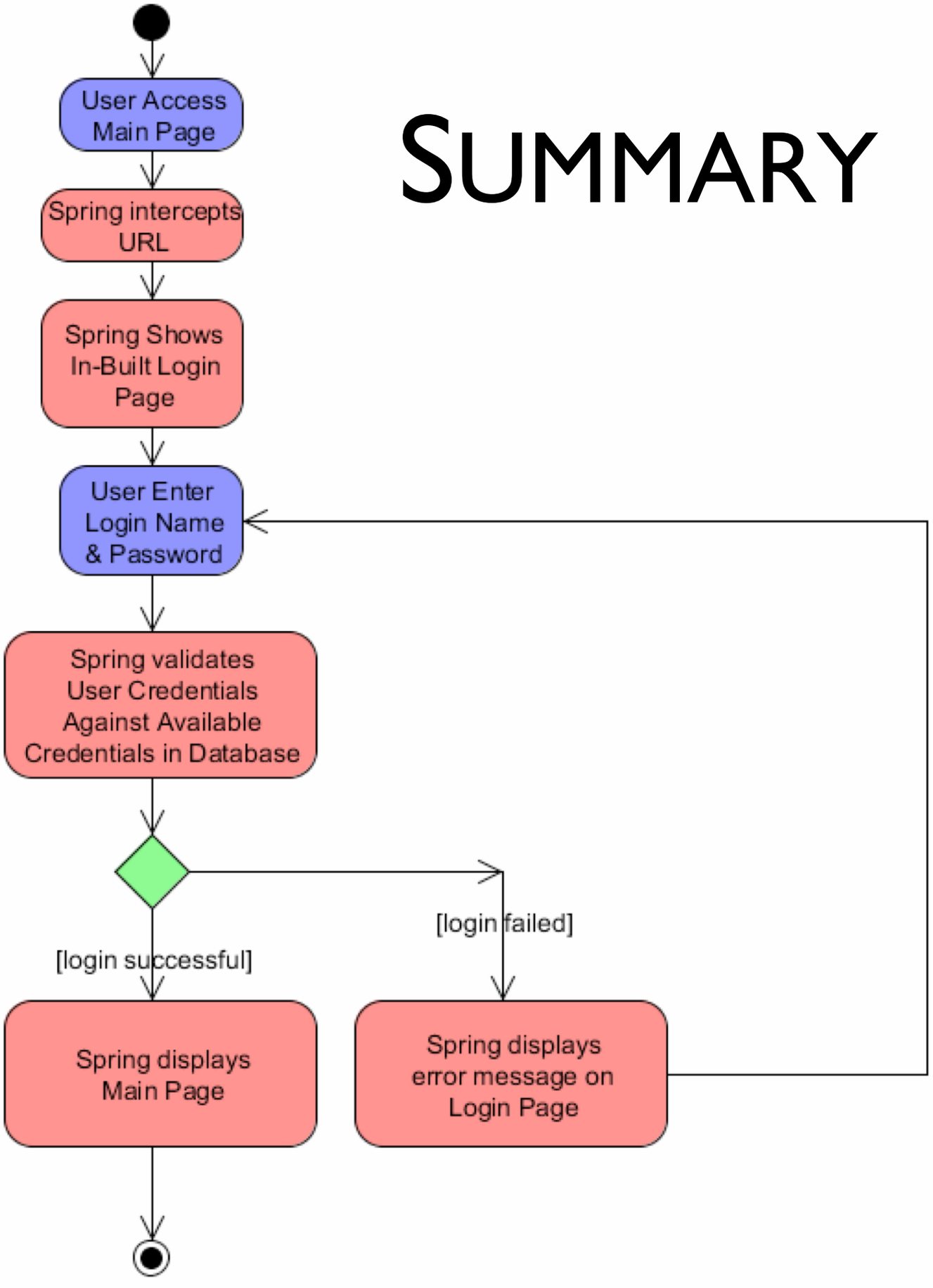
WebInitializer.java:

```
public class WebInitializer extends
                AbstractAnnotationConfigDispatcherServletInitializer {

    @Override
    protected Class<?>[] getRootConfigClasses() {
        return new Class[]{ApplicationConfig.class,
                           WebSecurityConfig.class };
    }

    ...
}
```

specify where to find
the security configuration

DEMO

SUMMARY


User Access Main Page
Spring intercepts URL
Spring Shows In-Built Login Page
User Enter Login Name & Password
Spring validates User Credentials Against Available Credentials in Database
[login successful]
[login failed]
Spring displays Main Page
Spring displays error message on Login Page

WHAT IF

I HAVE A LOT OF USERS ?

WebSecurityConfig.java:

use jdbc-user-service

```
@Autowired
private DataSource dataSource;

@Override
protected void configure(AuthenticationManagerBuilder auth) {

    auth.jdbcAuthentication().dataSource(dataSource)
        .usersByUsernameQuery("select username, password, enabled"
                + " from users where username=?")
        .authoritiesByUsernameQuery("select username, authority "
                + "from authorities where username=?")
        .passwordEncoder(new BCryptPasswordEncoder());
}
```

… or Create your own UserService

```
@Bean
public PasswordEncoder encoder() {
    return new BCryptPasswordEncoder(11);
}


@Bean
public UserDetailsService userDetailsService() {
    return new TourismUserService();
}


@Bean
public DaoAuthenticationProvider authProvider() {
    DaoAuthenticationProvider authProvider =
                                    new DaoAuthenticationProvider();
    authProvider.setUserDetailsService(userDetailsService());
    authProvider.setPasswordEncoder(new BCryptPasswordEncoder());
    return authProvider;
}
```

WHAT IF

I HAVE AN EXISTING LDAP SYSTEM ?

WebSecurityConfig.java:

```
@Override
protected void configure(AuthenticationManagerBuilder auth) {
    auth.ldapAuthentication()
      .contextSource().url("ldap://localhost:10389/dc=example,dc=com")
      .managerDn("uid=admin,ou=system").managerPassword("secret")
      .and()
      .userSearchBase("ou=users")
      .userSearchFilter("(cn={0})");
}
```

use ldap-authentication-provider

Other…

- OpenID Support
- Remember-Me Authentication
- HTTP/HTTPS Channel Security
- Custom login form
- Logout
- …

REFERENCES

- https://docs.spring.io/autorepo/docs/spring-security/current/reference/htmlsingle/