

Navigation

[Technical Articles](#)

[Home](#)

- [Home](#)

-

[Return to Content](#)

Registration – Activate a New Account by Email

[Elena](#) October 23, 2014 [Spring Security](#)

1. Overview

This article continues our ongoing Registration with Spring Security by finishing the missing piece of the registration process. Following this logic, a newly registered user will not be able to log in until email/registration verification is completed.

2. A VerificationToken Entity to Our Model associated to a . So, we need a c

We will make use of a simple verification token as the key artifact through which a user is verified.

2.1. Adding a VerificationToken entity must meet the following criteria:

The VerificationToken

There will be one VerificationToken User VerificationTokenUser

- It will be created after the user registration data is persisted.

- It will expire in 24 hours following initial registration.

Its value should be unique and randomly generated.

Requirements 2 and 3 are part of the registration logic. The other two are implemented in a simple VerificationToken entity. @Table public class VerificationToken { private static final int EXPIRATION =

2.2. Add an Enabled Flag to the User entity for now:

We will set the value of this flag depending on the result of the registration confirmation use case. Lets jus add the following

@Column(name = "enabled", private Boolean enabled;

3. The Account Registration Phase

Lets add two additional pieces of business logic to the user registration use case:

Generating a VerificationToken

- Sending the account confirmation email message which includes a confirmation link with the VerificationToken’

3.1. Using Spring Event Handling to Create the Token and Send the Verification E

These two additional pieces of logic should not be performed by the controller directly because they are “collateral” actions. @Autowired ApplicationEventPublisher @RequestMapping(value = "/user/registration", method

3.2. Spring Event Handler Implementation to start the that will handle the verificat

The controller is using an ApplicationEventPublisherRegistrationListener

An ApplicationEvent representing the completion of the user registration.

The RegistrationCompleteEvent is the only event that does not exist the controller will return a page. With the correspond

OnRegistrationCompleteEvent Example 3.2.2: OnRegistrationCompleteEvent extends ApplicationEvent

Here, the confirmRegistrationOnRegistrationCompleteEventUser implements ApplicationListener<OnRegistration

3.3. Processing the VerificationToken Parameter

When the user receives the “Confirm Registration” email, he will click on the attached link and fire a GET request. Example 3.3.1: RegistrationController If the controller will return a page. With the correspond

Notice that if there is no user associated with the VerificationTokenVerificationTokenBadUser.html

Example 3.3.2: If the token and user exist, the controller then proceeds to set the UserenabledVerificationToken @taglib prefix="sec" u

4. Adding Account Activation Checking to the Login Process

ladUserByUsername method:

Make sure the user is enabled before letting him log in.

Example 4.1: Shows the checking the VerificationToken in MyUserService with the flag set to false. This will trigger a

Notice that if the user is not enabled, the account is deleted and the method returns an org.springframework.security.core.

Now we need to modify our login.html to add Account Activation Error Checking

confirm test="\${param.error}"; if "\${param.error} eq 'Account Activation Error' choose

<c:when test="\${SPRING_SECURITY_

5. Adapting the Persistence Layer

We need to modify the API of the persistence layer by:

Creating a VerificationTokenRepository UserVerificationToken

Adding methods to the IUserService

VerificationTokenRepository Example 5.1: New interfaces and implementation:

UserService Example 5.2: “registerNewUser” method extends JpaRepository<VerificationToken, Long

UserService Example 5.3: IUserService User registerNewUserAccount(UserDto accountDto) t

UserService Example 5.4: class UserService implements IUserService { @Autowired private Use

6. Conclusion

We have expanded our Spring registration process to include an email based account activation procedure. The account a

Subscribe

Subscribe to our e-mail newsletter to receive updates.

[\(published\) Handling Static Resources With Spring](#)

[Convert HTML to PDF using Apache FOP](#)

No comments yet.

1 / 2014 Technical Articles All Rights Reserved

Leave a Reply [Click here to cancel reply.](#)

the footer should be inserted here?..

Logged in as [oDesk Authors](#). [Log out?](#)