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 proce
Following this logic, a newly registered user will not be able to log in until email/registration verification is completed.

2. A Verification Token Entity to Our Modelassociated 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.

Registration 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 followin

```
@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

```
@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 AplicationEvent representing the completion of the user registration.
- An ApplicationListener. For and access, and its implementation for new CRUID operations needed.

```
@Component public class RegistrationListener implements ApplicationListener<OnRegistrationCompleteEvent
```

Here, the confirmRegistrationOnRegistrationCompleteEventUser

3.3. Processing the Verification Token Parameter

When the user receives the “Confirm Registration” email, he will click on the attached link and fire a GET re

```
<%@ taglib prefix="sec" u
```

If the token and user exist, the controller then proceeds to set the UserenabledVerificationToken

4. Adding Account Activation Checking to the Login Process

ladUserByUsername method:

```
<c:if test="${param.error != null}">     <c:choose>     <c:when test="${SPRING_SECURIT
```

5. Adapting the Persistence Layer

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

- Creating a VerificationTokenRepository UserVerificationToken
- Adding methods to the IUserInterface

```
@Service public 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.

Leave a Reply Click here to cancel reply.

Logged in as odeskAuthor8. Log out?