Title: Spring Boot: How to specify the PasswordEncoder?

Post Body:

Currently I got the main class:

```
package com.recweb.springboot;  import org.springframework.boot.SpringApplication; import org.springframework.boot.autoconfigu
```

a members class (id, firstname..), a MemberController class:

```
package com.recweb.springboot;  import java.util.Arrays; import java.util.List;  import org.springframework.web.bind.annotatio
```

and a WebSecurityConfig class:

```
package com.recweb.springboot;  import org.springframework.context.annotation.Bean; import org.springframework.security.config
```

When i run "http://localhost:8080/members" i get a login page, i enter "user" as user & "user" as password and then i get the hardcoded Member. It worked fine, but then i right clicked on my project-Run as-Maven install (because i added a dependency, i don't know if that was necessary, first time with Maven too). Since then, when i enter "user" & "user" on the login page i get this error:

```
java.lang.IllegalArgumentException: There is no PasswordEncoder mapped for the id "null"     at org.springframework.security.c
```

and it stays on the login page. I tried to remove the dependency & Maven install again, but no luck. This is my POM:

```
<?xml version="1.0" encoding="UTF-8"?> <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XM
```

What went wrong? Thanks

Accepted Answer:

In `spring-security-core:5.0.0.RC1`, the default `PasswordEncoder` is built as a `DelegatingPasswordEncoder`. When you store the users in memory, you are providing the passwords in plain text and when trying to retrieve the encoder from the `DelegatingPasswordEncoder` to validate the password it can't find one that matches the way in which these passwords were stored.

Use this way to create users instead.

```
User.withDefaultPasswordEncoder().username('user').password('user').roles('USER').build();
```

You can also simply prefix `{noop}` to your passwords in order for the `DelegatingPasswordEncoder` use the `NoOpPasswordEncoder` to validate these passwords. Notice that `NoOpPasswordEncoder` is deprecated though, as it is not a good practice to store passwords in plain text.

```
User.withUsername('user').password('{noop}user').roles('USER').build();
```

For more information, check this post.

https://spring.io/blog/2017/11/01/spring-security-5-0-0-rc1-released#password-encoding

Highest Rated Answer:

Use `NoOpPasswordEncoder` for `inMemoryAuthentication`

```
auth.inMemoryAuthentication()     .withUser('user')     .password('{noop}password')     .roles('USER')
```