Injected MessageSource is null

Framework: Spring 3.

I really can't understand why the message source injectend in a bean ends up always to be NULL.

Here's the snippets:

the `servlet.xml`

```
<context:annotation-config />

<context:component-scan base-package="com.myproject.controllers" />

<mvc:annotation-driven />


<bean id="messageSource"
class="org.springframework.context.support.ReloadableResourceBundleMessageSource">
    <property name="basename" value="/WEB-INF/messages/messages" />
    <property name="cacheSeconds" value="0" />
</bean>
```

the class where the messageSource is injected

```
import com.myproject.controllers.forms.RegistrationForm;

@Component
public class RegistrationFormValidator implements Validator {

@Autowired
@Qualifier("messageSource")
private MessageSource messageSource;


    //other stuff here...

}
```

here's the controller

```
@Controller
@SessionAttributes("userSearchForm")
public class UsersController extends PaginationController<ProfiledUser>{


@InitBinder(value="registrationForm")
public void initBinder(WebDataBinder binder)
{
    binder.setValidator(new RegistrationFormValidator());
}
```

I have already tried the following:

1. deleting the annotations and injecting the message source via xml configuration file
2. implementing the MessageSourceAware interface
3. trying to inject a `ReloadableresourceBundleMessageSource` instead of using interface `MessageSource`

everything ends up in a epic fail ;-) How can I get the MessageSource properly injected?

java spring

edited Nov 24 '12 at 16:44

asked Nov 24 '12 at 14:44

MaVVamaldo
1,555 3 19 44

may be related: stackoverflow.com/questions/4567512/… – user180100 Nov 24 '12 at 15:13

Do you have an other `component-scan` in an other spring context? – Ralph Nov 24 '12 at 15:31

@Ralph - yes, I do. I have two component-scan in a separate context file but they don't conflict, as far as I know. – MaVVamaldo Nov 24 '12 at 15:36

How do you know that it's really null? @Autowired requires the field to be populated - does Spring print the exception stack trace during initialization which includes `org.springframework.beans.factory.BeanCreationException: Injection of autowired dependencies failed for class` ? How is your **Validator** instance created, with the `new` operator? – Boris Treukhov Nov 24 '12 at 15:51

@Boris Treukhov - no, the exception isn't thrown. While debugging I can see that the messageSource attribute for the class RegistrationFormValidator is simply null. I argue the injection went wrong because I can use the messageSource declared in the servlet.xml file elsewhere. For example while resolving error messages in different forms. I know spring does it automatically, so I can't be sure about the rationale laying behind the courtains. – MaVVamaldo Nov 24 '12 at 15:56

4 Answers

~~I think that you are examining the field values of a CGLIB class~~

~~See spring singleton bean fields are not populated~~ **update**

A general note about autowiring

`@Autowired` annotation is processed by `AutowiredAnnotationBeanPostProcessor` which can be registered by specifying `<context:annotation-config />` annotation in the respective spring configuration file(bean postprocessors work on a per container basis so you need to have different postprocessors for the servlet and for the application root context => you need to put `<context:annotation-config />` both to the web app context and the dispatcher servlet configuration).

Please note that `@Autowired` annotation has `required` property which is set as default to true, that means that if the autowiring process occurs Spring will check that exactly one instance of the specified bean exists. If the bean which fields are annotated with `@Autowired` is a singleton then, the check will be performed during the application initialization.

update In this specific question the `Validator` instance was not created by Spring at all, that is why no autowiring was performed and no initialization exceptions were thrown.

edited May 23 '17 at 10:28
Community ♦
1 1

answered Nov 24 '12 at 15:58
Boris Treukhov
12.3k 4 48 74

change basename to something like this:

```
<property name="basename" value="classpath:messages/messages" />
```

answered Mar 6 '14 at 15:03
Breandán Dalton
1,189 8 18

In the java configs class add the following beans.

```
@Bean
public MessageSource messageSource() {
    ResourceBundleMessageSource messageSource = new ResourceBundleMessageSource();
    messageSource.setBasename("messages");

    return messageSource;
}

@Bean
public LocaleResolver localeResolver() {
    SessionLocaleResolver resolver = new SessionLocaleResolver();
    resolver.setDefaultLocale(Locale.ENGLISH);

    return resolver;
}

@Bean
public MessageSourceAccessor messageSourceAccessor(MessageSource messageSource){
    return new MessageSourceAccessor(messageSource, Locale.ENGLISH );
}
```

Then the `MessageSourceAccessor` bean can be injected as follows

```
@Autowired
private MessageSourceAccessor msgs;
```

Get message strings as follows,

```
msgs.getMessage("controller.admin.save.success")
```

The `message_en.properties` file should be inside `/src/main/resources` folder. Add necessary properties file for other languages as well.

Hope this helps.

answered Aug 18 '16 at 14:14

tharindu_DG

3,775 3 25 34

I had similar issue when I tried to use the MessageSource in a custom Date Formatter.

The code AppDateFormatter

```
public class AppDateFormatter implements Formatter<Date> {

    @Autowired
    private MessageSource messageSource;

    other stuff.....

    private SimpleDateFormat createDateFormat(final Locale locale) {
        final String format = this.messageSource.getMessage("date.format", null, locale);
        final SimpleDateFormat dateFormat = new SimpleDateFormat(format);
        dateFormat.setLenient(false);
       return dateFormat;
   }

}
```

This is what worked for me :

```
public class WebMvcConfiguration extends WebMvcConfigurerAdapter {

  other stuff.....

    @Bean
    public MessageSource messageSource() {
        ResourceBundleMessageSource messageSource = new ResourceBundleMessageSource();
        messageSource.setBasenames("Messages/Messages", "Messages/Labels");
        messageSource.setDefaultEncoding("UTF-8");
        messageSource.setCacheSeconds(1);
        return messageSource;
    }

    @Bean
    public AppDateFormatter appDateFormatter(){
        return new AppDateFormatter();
    }

    @Bean
    public FormattingConversionService mvcConversionService() {
        FormattingConversionService conversionService = new
DefaultFormattingConversionService();
        conversionService.addFormatter(appDateFormatter());
        return conversionService;
    }

}
```

answered Oct 30 '14 at 6:39

Gauri Telang

1 1