How to configure multiple (unknown amount) different smtp servers in spring boot?

Asked 1 year ago Modified 10 months ago Viewed 240 times

0

My question looks similar to how to configure two different emails in springboot?

And this answer is quite cool But it differs from my needs. I don't know how many smtp servers will be configured. I want to have ability to add email servers only by adding several lines to configuration.

In the code I want to @Autowire them like:

```
@Autowire
private List<JavaMailSenderImpl> javaMailSenders
```

I want to have config like

```
spring.mail.first.host=host1
spring.mail.first.port=port1
spring.mail.first.username=username1
spring.mail.first.password=password1
spring.mail.first.properties.mail.smtp.auth=true
...
spring.mail.second.host=host2
spring.mail.second.port=port2
spring.mail.second.username=username2
spring.mail.second.password=password2
spring.mail.second.properties.mail.smtp.auth=true
...
spring.mail.fifth.host=host5
spring.mail.fifth.port=port5
spring.mail.fifth.username=username5
spring.mail.fifth.password=password5
spring.mail.fifth.properties.mail.smtp.auth=true
```

Could you please help ?

java spring spring-boot email smtp

Share Edit Follow

edited Jun 20, 2023 at 13:29

asked Jun 16, 2023 at 16:32
gstackoverflow
36.4k ● 130 ● 397 ● 758

2 Answers

Sorted by: Highest score (default)

You can combine the original answer with this answer about creating beans dynamically.

2

For example, let's say you create your own `MailListProperties`:

```
@ConfigurationProperties(prefix = "mail")
public record MailListProperties(Map<String, MailProperties> list) { }
```

Then you can create your own `BeanDefinitionRegistryPostProcessor` class to create the `JavaMailSenderImpl` beans:

```
public class DynamicMailsenderBeanDefinitionRegistrar implements
BeanDefinitionRegistryPostProcessor {
    private final MailListProperties properties;

    public DynamicMailsenderBeanDefinitionRegistrar(Environment environment) {
        this.properties = Binder
            .get(environment)
            .bind("mail", MailListProperties.class)
            .orElseThrow(IllegalStateException::new);
    }

    @Override
    public void postProcessBeanDefinitionRegistry(BeanDefinitionRegistry
registry) throws BeansException {
        properties.list().forEach((beanName, mailProperties) -> {
            GenericBeanDefinition definition = new GenericBeanDefinition();
            definition.setBeanClass(JavaMailSenderImpl.class);
            definition.setInstanceSupplier(() -> {
                JavaMailSenderImpl sender = new JavaMailSenderImpl();
                sender.setHost(mailProperties.getHost());
                // TODO: Here you put the logic to create a JavaMailSenderImpl
                // You can read the properties defined in MailProperties
                return sender;
            });
            registry.registerBeanDefinition(beanName, definition);
        });
    }

    @Override
    public void postProcessBeanFactory(ConfigurableListableBeanFactory
beanFactory) throws BeansException {

    }
}
```

To register the `DynamicMailsenderBeanDefinitionRegistrar` bean, you can use the following code:

```
@Bean
public DynamicMailsenderBeanDefinitionRegistrar registrar(Environment
environment) {
    return new DynamicMailsenderBeanDefinitionRegistrar(environment);
}
```

To configure the properties, you can use the following properties file:

```
mail.list.first.host=...
mail.list.first.username=...
mail.list.first.password=...
mail.list.second.host=...
mail.list.second.username=...
mail.list.second.password=...
mail.list.third.host=...
mail.list.third.username=...
mail.list.third.password=...
```

In this example, the keys (`first`, `second`, and `third`) in the properties file will become the bean names. You can choose to autowire a specific JavaMailSenderImpl bean like this:

```
@Autowired
@Qualifier("first")
private JavaMailSenderImpl firstMailSender;
```

Alternatively, you can autowire all the beans of a specific type using a collection:

```
@Autowired
private List<JavaMailSenderImpl> mailSenders;
```

Share Edit Follow   edited Aug 29, 2023 at 20:37   answered Jun 19, 2023 at 8:21
Dimitri Mestdagh
43.9k ● 15 ● 104 ● 140

Hi, just a code fix: registry.registerBeanDefinition(**key**, definition); **key** is undefined Simply use registry.registerBeanDefinition(**beanName**, definition); Finally, you can also use: @Autowired private Map<String, JavaMailSenderImpl> beanDefinitionMap; – Tama Aug 29, 2023 at 14:45

0

Why not save the email configuration in the database?

If you must put the configuration in properties you can do like this

```
spring.mail.host=host1,host2
spring.mail.port=port1,port2
spring.mail.username=username1,username2
spring.mail.password=password1,password2
spring.mail.properties.mail.smtp.auth=true,true
```

by this you can you avoid changing Java code when you change the properties, but you need to define a separator to separate different hosts like `,` or something else then prase in code.

I think the second method is hard to maintain Configuration.

Share Edit Follow   answered Jun 17, 2023 at 7:55
lant
488 ● 3 ● 11