How to make HibernateValidator pick up Locale from HTTP request Header for measage templates in spring Boot?

Asked 1 year, 2 months ago Modified 1 year, 2 months ago Viewed 331 times

1

I've been cracking my head on this problem for a week. After deep research and debugging session I'm just desperate.

All want to do is for the `HibernateValidator` to pick up messages from the provided `MessageSource` and use locale to interpolate messages from the `MessageSource` according to the template provided in the annotation for the object field.

So far the only thing `HibernateValidator` does is falling back to the `Locale.getDefault()` every time no matter what I do.

This thing where Spring developers say that there is to easy way to do that.

https://github.com/spring-projects/spring-boot/issues/20673

This is the topic that describes the setup I should use to get the result that still does not work.

Spring Boot, Hibernate validator language based on LocaleContextHolder

This is my starting setup that just work with one default locale in my case.

What do I need to add or change in order for the `HibernateValidator` to pick up `Locale` from the Header and interpolate messages from the `MessageSource` in the correct `Locale` picked up from the request header?

Here is the primitive setup I've got.

Controller.

```
package com.example.gatewayservice.controller;

@RestController
@RequiredArgsConstructor
@Slf4j
@RequestMapping("/api/v1")
//@Validated
public class UserDataController {

    @Autowired
    UserDataService userDataService;

    @Async("threadPoolTaskExecutor")
    @PostMapping("/getSimData")
//    @Validated(OnRequestValidation.class)
    public CompletableFuture<ResponseEntity<SimDataResponse>> getSimData(@Valid
@RequestBody SimDataRequest request) {
```

```
        log.info("Entered getSimData controller with the request ::\n {}",
request);

        ResponseEntity<SimDataResponse> result =
ResponseEntity.ok(userDataService.getSimData(request));

        return CompletableFuture.completedFuture(result);
    }
}
```

Request object that is validated with Hibernate validator.

```
package com.example.gatewayservice.dto.controller;

@Data
public class SimDataRequest {

    @NotEmpty(message = "{msisdn.error.empty}")
    private String msisdn;

}
```

messages.properties file content

```
msisdn.error.empty="MSISDN is empty"
```

messages_fr.properties file content

```
msisdn.error.empty=Le MSISDN est vide
```

And the root of all problems, the configuration that for the love of me I can't figure out.

```
package com.example.gatewayservice.config;

@Configuration
public class ServiceConfig implements WebMvcConfigurer {

    @Bean
    public RestTemplate restTemplate(RestTemplateBuilder restTemplateBuilder) {

        OkHttpClient okHttpClient = new OkHttpClient.Builder()
                .connectTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .build();

        return restTemplateBuilder
                .requestFactory(() -> new
OkHttp3ClientHttpRequestFactory(okHttpClient))
                .build();

    }

    @Bean("threadPoolTaskExecutor")
    public TaskExecutor asyncExecutor() {
        ThreadPoolTaskExecutor executor = new ThreadPoolTaskExecutor();
        executor.setCorePoolSize(50);
```

```
        executor.setMaxPoolSize(1000);
        executor.setQueueCapacity(50);
        executor.setWaitForTasksToCompleteOnShutdown(true);
        executor.setThreadNamePrefix("Async-");
        return executor;
    }



    /*locale resolution strategy*/
    @Bean
    public AcceptHeaderLocaleResolver localeResolver() {
        final CustomLocaleResolver resolver = new CustomLocaleResolver();
        resolver.setSupportedLocales(Arrays.asList(Locale.FRANCE, Locale.US,
Locale.UK));
        resolver.setDefaultLocale(Locale.US);
        return resolver;
    }

    public static class CustomLocaleResolver extends AcceptHeaderLocaleResolver {
        List<Locale> LOCALES = Arrays.asList(
                new Locale("en"),
                new Locale("fr"));
        @Override
        public Locale resolveLocale(HttpServletRequest request) {
            String headerLang = request.getHeader("Accept-Language");
            return headerLang == null || headerLang.isEmpty()
                    ? Locale.getDefault()
                    : Locale.lookup(Locale.LanguageRange.parse(headerLang),
LOCALES);
        }
    }

    /*default message source*/
    @Bean
    public MessageSource messageSource() {
        ReloadableResourceBundleMessageSource messageSource = new
ReloadableResourceBundleMessageSource();
//        ResourceBundleMessageSource messageSource = new
ResourceBundleMessageSource();
        messageSource.setBasename("messages");
        messageSource.setDefaultEncoding("UTF-8");
        return messageSource;
    }

    /*Validator setup*/
    @Bean
    public LocalValidatorFactoryBean validator() {
        LocalValidatorFactoryBean factoryBean = new LocalValidatorFactoryBean();

        factoryBean.setValidationMessageSource(messageSource());

//        MessageInterpolatorFactory interpolatorFactory = new
MessageInterpolatorFactory(validationMessageSource());
//        factoryBean.setMessageInterpolator(interpolatorFactory.getObject());

        return factoryBean;
    }

}
```

java spring-boot locale hibernate-validator

Share Edit Follow

asked Apr 24, 2023 at 10:20
Roman Y
53 ● 1 ● 3

1 Answer

Sorted by: Highest score (default)

2

You might want to take a look at the `LocaleResolver` provided by Hibernate Validator.

You could create a resolver that would work based on the `LocaleContextHolder` which has its locale set by the `RequestContextFilter` :

```
@Bean
public LocalValidatorFactoryBean validator() {
    LocalValidatorFactoryBean factoryBean = new LocalValidatorFactoryBean();

    factoryBean.setValidationMessageSource(messageSource());
    factoryBean.setConfigurationInitializer( configuration -> {
        // cast to Hibernate Validator specific configuration to be able to
access
        // the locale resolver:
        ( (HibernateValidatorConfiguration) configuration ).localeResolver(
                new LocaleResolver() {
                    @Override
                    public Locale resolve(LocaleResolverContext context) {
                        // use the locale from the context holder.
                        // this works with RequestContextFilter.
                        return LocaleContextHolder.getLocale();
                    }
                }
        );
    });
    return factoryBean;
}
```

If the locale resolution by that filter doesn't meet your needs, you can create your own filter and add it to the chain. The general idea remains the same - get the locale in the filter and set it to some thread local variable, and retrieve the locale in the `LocaleResolver`.

Share Edit Follow

answered Apr 24, 2023 at 13:55
mark_o
2,438 ● 1 ● 13 ● 19