Application does not start due to two bean instances of org.springframework.plugin.core.PluginRegistry (Follow-up) #1094

New issue

Closed **rainer198** opened this issue on 17 Oct 2019 · 19 comments

rainer198 commented on 17 Oct 2019 • edited

As explained in the last comment of #966 , an application using `spring-hateoas-1.0.0` does not start if the bean factory is setup in a way that the `ParameterNameDiscoverer` is null or an alternative implementation.

The reason is that when autowiring of `PluginRegistry` into the bean `_relProvider` it finds two candidates and choosing one soley relies on the constructor argument name (same in `EntityLinksConfiguration` where the other instance of `PluginRegistry` is created.)

This could be easily resolved by using `@Qualifier` annotations and named beans.

👍 4

7fantasy7 commented on 17 Oct 2019 • edited

Having very similar issue after upgrading spring boot to 2.2.0

APPLICATION FAILED TO START

Description:

Parameter 0 of method linkDiscoverers in org.springframework.hateoas.config.HateoasConfiguration required a single bean, but 17 were found:
- modelBuilderPluginRegistry: defined in null
- modelPropertyBuilderPluginRegistry: defined in null
- typeNameProviderPluginRegistry: defined in null
- syntheticModelProviderPluginRegistry: defined in null
- documentationPluginRegistry: defined in null
- apiListingBuilderPluginRegistry: defined in null
- operationBuilderPluginRegistry: defined in null
- parameterBuilderPluginRegistry: defined in null
- expandedParameterBuilderPluginRegistry: defined in null
- resourceGroupingStrategyRegistry: defined in null
- operationModelsProviderPluginRegistry: defined in null
- defaultsProviderPluginRegistry: defined in null
- pathDecoratorRegistry: defined in null
- apiListingScannerPluginRegistry: defined in null
- relProviderPluginRegistry: defined by method 'relProviderPluginRegistry' in org.springframework.hateoas.config.HateoasConfiguration
- linkDiscovererRegistry: defined in null
- entityLinksPluginRegistry: defined by method 'entityLinksPluginRegistry' in class path resource [org/springframework/hateoas/config/WebMvcEntityLinksConfiguration.class]

Action:

Consider marking one of the beans as **@primary**, updating the consumer to accept multiple beans, or using **@qualifier** to identify the bean that should be consumed

danieleforberghi commented on 22 Oct 2019

Agree using spring-boot-starter-hateoas version 2.2.0.RELEASE get same error. Please fix this!

org.springframework.data spring-data-releasetrain Moore-RELEASE import pom
and springfox 2.9.2

mgargadennec commented on 25 Oct 2019

Same here after migration to Spring-Boot 2.2.0.RELEASE.

👍 3

chadmalla-ff commented on 25 Oct 2019

> Having very similar issue after upgrading spring boot to 2.2.0
>
> APPLICATION FAILED TO START
>
> Description:
>
> Parameter 0 of method linkDiscoverers in org.springframework.hateoas.config.HateoasConfiguration required a single bean, but 17 were found:
>
> - modelBuilderPluginRegistry: defined in null
> - modelPropertyBuilderPluginRegistry: defined in null
> - typeNameProviderPluginRegistry: defined in null
> - syntheticModelProviderPluginRegistry: defined in null
> - documentationPluginRegistry: defined in null
> - apiListingBuilderPluginRegistry: defined in null
> - operationBuilderPluginRegistry: defined in null
> - parameterBuilderPluginRegistry: defined in null
> - expandedParameterBuilderPluginRegistry: defined in null

Assignees
No one assigned

Labels
resolution: invalid

Projects
None yet

Milestone
No milestone

11 participants

- resourceGroupingStrategyRegistry: defined in null
- operationModelsProviderPluginRegistry: defined in null
- defaultsProviderPluginRegistry: defined in null
- pathDecoratorRegistry: defined in null
- apiListingScannerPluginRegistry: defined in null
- relProviderPluginRegistry: defined by method 'relProviderPluginRegistry' in org.springframework.hateoas.config.HateoasConfiguration
- linkDiscovererRegistry: defined in null
- entityLinksPluginRegistry: defined by method 'entityLinksPluginRegistry' in class path resource [org/springframework/hateoas/config/WebMvcEntityLinksConfiguration.class]

Action:

Consider marking one of the beans as **@primary**, updating the consumer to accept multiple beans, or using **@qualifier** to identify the bean that should be consumed

+1 Would like to a see a solution for this error please

chadmalla-ff commented on 25 Oct 2019

The workaround I have at the moment is to remove all `springfox` related dependencies in the project and I no longer get the above error.

👍 1

keshavsharma-github commented on 25 Oct 2019

@chadmalla-ff also facing the same issue.
Error starting ApplicationContext. To display the conditions report re-run your application with 'debug' enabled.
2019-10-25 12:06:23.321 ERROR 14400 --- [restartedMain] o.s.b.d.LoggingFailureAnalysisReporter :

APPLICATION FAILED TO START

Description:

Parameter 0 of method linkDiscoverers in org.springframework.hateoas.config.HateoasConfiguration required a single bean, but 16 were found:
- modelBuilderPluginRegistry: defined in null
- modelPropertyBuilderPluginRegistry: defined in null
- typeNameProviderPluginRegistry: defined in null
- documentationPluginRegistry: defined in null
- apiListingBuilderPluginRegistry: defined in null
- operationBuilderPluginRegistry: defined in null
- parameterBuilderPluginRegistry: defined in null
- expandedParameterBuilderPluginRegistry: defined in null
- resourceGroupingStrategyRegistry: defined in null
- operationModelsProviderPluginRegistry: defined in null
- defaultsProviderPluginRegistry: defined in null
- pathDecoratorRegistry: defined in null
- apiListingScannerPluginRegistry: defined in null
- relProviderPluginRegistry: defined by method 'relProviderPluginRegistry' in class path resource [org/springframework/hateoas/config/HateoasConfiguration.class]
- linkDiscovererRegistry: defined in null
- entityLinksPluginRegistry: defined by method 'entityLinksPluginRegistry' in class path resource [org/springframework/hateoas/config/WebMvcEntityLinksConfiguration.class]

Action:

Consider marking one of the beans as **@primary**, updating the consumer to accept multiple beans, or using **@qualifier** to identify the bean that should be consumed.

Please provide inputs if you find any solution to this issue.

danieleforberghi commented on 25 Oct 2019

@chadmalla-ff it is not an option to remove all springfox deps

danieleforberghi commented on 25 Oct 2019

@rainer198 Understand making beans primary or such. But which beans to override from HateoasConfiguration?

gregturn commented on 25 Oct 2019 Member

The community has identified springfox on the classpath as the key factor. Thanks for doing that.

How do you propose we test any potential changes? We don't want to add springfox to this project since this project does nothing with that project.

danieleforberghi commented on 25 Oct 2019 • edited

@gregturn Understand it should be the springfox community who is responsible to be compatible with latest Spring Boot 2.2.0.RELEASE and it's subprojects. But while googling this it appears in your community.

danieleforberghi commented on 25 Oct 2019

@gregturn The reason to the error is that springfox uses an earlier version of spring-plugin which is not compatible with spring-hateoas which uses 2.0.0.RELEASE of the plugin. Tried to workaround this but realized it is not possible until springfox releases a new version using the latest spring-plugin version.

gregturn added the `resolution: invalid` label on 29 Oct 2019

gregturn commented on 29 Oct 2019 Member

I'm going to label this as invalid, since the real issue is that Spring HATEAOS is not presently compatible with Springfox (based on older versions of Spring Plugin and Spring Framework).

Feel free to track efforts on Springfox.

gregturn closed this on 29 Oct 2019

---

rainer198 commented on 30 Oct 2019 — Author

Sorry, have not been in office for a couple of days and missed the discussion here.

I do not agree that this is related to spring fox. I don't have spring fox on the classpath and can reproduce the issue.

The issue is that Spring HATEOAS relies on Spring autowiring fallback. If multiple candidate beans are found (here: multiple PluginRegistry beans) to be injected into another bean as constructor argument (here: relProvider), then Spring tries to guess the correct canidate by comparing bean names with constructor argument name. If any canidate matches this one is chosen.

However, it is possible to turn off this fallback in a spring application. One way is to set the `ParameterNameDiscoverer` to null in a `BeanFactoryPostProcessor`

```
@Override
public void postProcessBeanFactory(final ConfigurableListableBeanFactory beanFactory) {
        AbstractAutowireCapableBeanFactory factory = (AbstractAutowireCapableBeanFactory
        factory.setParameterNameDiscoverer(null);
}
```

Doing this, the fallback mechanism is not able to resolve constructor arguments and can't decide for a candidate. Instead, an exception is thrown.

Turning off the fallback is a valid configuration in our opinion if you want to avoid autowiring the wrong instance of a bean by chance. The term "fallback" is also used by Spring (see `org.springframework.beans.factory.support.DefaultListableBeanFactory`, line 1537 .

Fixing this on Spring HATEOAS side is pretty simple. Each PluginRegistry bean should have explicit names (e.g. `@Bean(name= relProviderPluginRegistry`) and each constructor argument of type PluginRegistry should refer to exactly one of these names (e.g. `@Qualifier("relProviderPluginRegistry")`

👍 2

---

edgarmariorivera commented on 11 Nov 2019

Hi guys, does anyone has a workaround for this? I would like to update my springboot app to 2.2.1 so we can use delayed instantiation feature.

Thanks

---

gregturn commented on 11 Nov 2019 — Member

@rainer198 That's an awfully quirky solution. It implies that you shouldn't use autowiring anywhere.

> If multiple candidate beans are found (here: multiple PluginRegistry beans) to be injected into another bean as constructor argument (here: relProvider), then Spring tries to guess the correct canidate by comparing bean names with constructor argument name.

Take away Spring HATEOAS and instead look at any Spring application, and explain why we shouldn't presume we can do this.

If there are two `PluginRegistry` beans, that sounds like a problem in and of itself. Why do you need more than one?

---

rainer198 commented on 12 Nov 2019 — Author

> If there are two PluginRegistry beans, that sounds like a problem in and of itself. Why do you need more than one?

It's not me needing more than one but you ;-)

1. One is defined in `org.springframework.hateoas.config.HatoasConfiguration::relProviderPluginRegistry()` consumed **only** in `_relProvider(...)` in the same configuration class.
2. another one on `org.springframework.hateoas.config.EntityLinksConfiguration::entityLinksPluginRegistry()` which is **only** consumed/autowired in `delegatingEntityLinks(..)` in the same configuration class

So, Spring HATEOAS instantiates two PluginRegistry beans, and the only way autowiring works here is based on a fallback based on parameter names. In our application this fallback has been turned off for reasons (we have a lot of beans of same interface, in our application and we wanted to force developers to explicitly state in a constructor which bean to use.)

---

laran commented on 18 Nov 2019 • edited

I have a similar issue. But I run into it when using spring-cloud-dataflow-rest-client and spring-boot-starter-test together.

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <version>2.2.1.RELEASE</version>
        <scope>test</scope>
        <exclusions>
                <exclusion>
                        <groupId>org.junit.vintage</groupId>
                        <artifactId>junit-vintage-engine</artifactId>
                </exclusion>
        </exclusions>
</dependency>
```

```
<dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-dataflow-rest-client</artifactId>
        <version>2.2.1.RELEASE</version>
</dependency>
```

It seems to be something in spring-boot-starter-test that automatically loads all beans of a type.

---

chrisgleissner commented 27 days ago

Hi,

I am facing the same issue after trying to upgrade a project that relies on both Springfox and HATEOAS from Spring Boot 2.1 to Spring Boot 2.2. I see that **@rainer198** suggested a solution within HATEOAS on October 30th. The ticket was then closed and the discussion fizzled out. But the problem remains.

Could **@rainer198** 's suggestion be incorporated into HATEOAS, **@gregturn**, or do you have any other suggestion on how this could be resolved (other than not upgrading at all or removing one of HateOAS or Springfox)? If I am forced to remove a library, it will most likely be HATEOAS as Springfox provides essential self-documentation of the REST API in the project.

Thanks

chrisgleissner mentioned this issue 27 days ago

Does not run in spring-boot 2.2.0 #5 Open

domselvon commented 16 days ago

I'm in the same boat as **@chrisgleissner**. If there's any chance this can be addressed in HateOAS as per **@rainer198** 's suggestion that woul dbe great. Otherwise, I'll also have to remove HateOAS from my application for the same reasons as **@chrisgleissner**.

Thanks in hopefulness.