Issue when using Swagger latest version 2.9.2 with Spring boot 2.2.0 #2932

New issue

Closed **AdityaJaiswal7** opened this issue on 8 Mar 2019 · 94 comments

AdityaJaiswal7 commented on 8 Mar 2019

Hi, I am running a Spring boot 2.2.0 with swagger version **2.9.2** and in swagger configuration file if I am using **@configuration** and **@EnableSwagger2**, getting the following error:

APPLICATION FAILED TO START

Description:

An attempt was made to call a method that does not exist. The attempt was made from the following location:

```
springfox.documentation.spring.web.plugins.DocumentationPluginsManager.createContextBuil
```

The following method did not exist:

```
org.springframework.plugin.core.PluginRegistry.getPluginFor(Ljava/lang/Object;Lorg/spring
```

The method's class, org.springframework.plugin.core.PluginRegistry, is available from the following locations:

```
jar:file:/Users/SystemName/.m2/repository/org/springframework/plugin/spring-plugin-core/
```

It was loaded from the following location:

```
file:/Users/SystemName/.m2/repository/org/springframework/plugin/spring-plugin-core/2.0.
```

Action:

Correct the classpath of your application so that it contains a single, compatible version of org.springframework.plugin.core.PluginRegistry

Any help would be appreciated.
Thanks!

👍 81 ❤️ 7

Assignees
No one assigned

Labels
maintenance

Projects
None yet

Milestone
3.0

65 participants

and others

N4zroth commented on 11 Mar 2019

I think you need to use springfox 3.0-SNAPSHOT to get it to work with the latest version of spring boot.

cer commented on 17 Mar 2019

Springfox 3.0.0-SNAPSHOT appears to not fix the problem with springBootVersion=2.2.0.M1

ghostd commented on 18 Mar 2019

It seems the `PluginRegistry#getPluginFor(S delimiter, T plugin)` method is now `PluginRegistry#getPluginOrDefaultFor(S delimiter, T plugin)`

vargan24 commented on 18 Mar 2019

Temporary solution:
Create 2 primary beans as following:
public class DocumentationPluginsManagerBootAdapter extends DocumentationPluginsManager {
@Autowired
@qualifier("resourceGroupingStrategyRegistry")
private PluginRegistry<ResourceGroupingStrategy, DocumentationType> resourceGroupingStrategies;

```
@Autowired
@Qualifier("defaultsProviderPluginRegistry")
private PluginRegistry<DefaultsProviderPlugin, DocumentationType> defaultsProviders;

@Override
public ResourceGroupingStrategy resourceGroupingStrategy(DocumentationType documentation
        return resourceGroupingStrategies.getPluginOrDefaultFor(documentationType, new S
}

@Override
public DocumentationContextBuilder createContextBuilder(DocumentationType documentationT
                DefaultConfiguration defaultConfiguration) {
        return defaultsProviders.getPluginOrDefaultFor(documentationType, defaultConfigu
                        .create(documentationType).withResourceGroupingStrategy(resource
}
```

}

public class TypeNameExtractorBootAdapter extends TypeNameExtractor {
private final TypeResolver typeResolver;

private final PluginRegistry<TypeNameProviderPlugin, DocumentationType> typeNameProviders;
private final EnumTypeDeterminer enumTypeDeterminer;

```
public TypeNameExtractorBootAdapter(TypeResolver typeResolver,
                PluginRegistry<TypeNameProviderPlugin, DocumentationType> typeNameProvide
                EnumTypeDeterminer enumTypeDeterminer) {
        super(typeResolver, typeNameProviders, enumTypeDeterminer);
        this.typeResolver = typeResolver;
        this.typeNameProviders = typeNameProviders;
        this.enumTypeDeterminer = enumTypeDeterminer;
}

public String typeName(ModelContext context) {
        ResolvedType type = asResolved(context.getType());
        if (isContainerType(type)) {
                return containerType(type);
        }
        return innerTypeName(type, context);
}

private ResolvedType asResolved(Type type) {
        return typeResolver.resolve(type);
}

private String genericTypeName(ResolvedType resolvedType, ModelContext context) {
        Class<?> erasedType = resolvedType.getErasedType();
        GenericTypeNamingStrategy namingStrategy = context.getGenericNamingStrategy();
        ModelNameContext nameContext = new ModelNameContext(resolvedType.getErasedType(),
                        context.getDocumentationType());
        String simpleName = fromNullable(typeNameFor(erasedType)).or(typeName(nameContext
        StringBuilder sb = new StringBuilder(String.format("%s%s", simpleName, namingStra
        boolean first = true;
        for (int index = 0; index < erasedType.getTypeParameters().length; index++) {
                ResolvedType typeParam = resolvedType.getTypeParameters().get(index);
                if (first) {
                        sb.append(innerTypeName(typeParam, context));
                        first = false;
                } else {
                        sb.append(String.format("%s%s", namingStrategy.getTypeListDelimi
                                        innerTypeName(typeParam, context)));
                }
        }
        sb.append(namingStrategy.getCloseGeneric());
        return sb.toString();
}

private String innerTypeName(ResolvedType type, ModelContext context) {
        if (type.getTypeParameters().size() > 0 && type.getErasedType().getTypeParameter
                return genericTypeName(type, context);
        }
        return simpleTypeName(type, context);
}

private String simpleTypeName(ResolvedType type, ModelContext context) {
        Class<?> erasedType = type.getErasedType();
        if (type instanceof ResolvedPrimitiveType) {
                return typeNameFor(erasedType);
        } else if (enumTypeDeterminer.isEnum(erasedType)) {
                return "string";
        } else if (type instanceof ResolvedArrayType) {
                GenericTypeNamingStrategy namingStrategy = context.getGenericNamingStrat
                return String.format("Array%s%s%s", namingStrategy.getOpenGeneric(),
                                simpleTypeName(type.getArrayElementType(), context), nam
        } else if (type instanceof ResolvedObjectType) {
                String typeName = typeNameFor(erasedType);
                if (typeName != null) {
                        return typeName;
                }
        }
        return typeName(new ModelNameContext(type.getErasedType(), context.getDocumentat
}

private String typeName(ModelNameContext context) {
        TypeNameProviderPlugin selected = typeNameProviders.getPluginOrDefaultFor(contex
                        new DefaultTypeNameProvider());
        return selected.nameFor(context.getType());
}
```

}

👍 11 🚀 4

abitgen commented on 24 Mar 2019

Added this to dependency and it worked for me.

```
        <dependency>
            <groupId>org.springframework.plugin</groupId>
            <artifactId>spring-plugin-core</artifactId>
            <version>1.2.0.RELEASE</version>
        </dependency>
```

👍 43 👎 2 🎉 7 ❤️ 18 🚀 7

burberius mentioned this issue on 25 Mar 2019

Snapshot 3.0.0 broken. @ConditionalOnClass did not find required class Open
#2929

NarasimhaMuddunri commented on 6 Apr 2019

Hi,

Am facing the same issue #2929.

Please let me know if any alternative.

Thanks in Advance !!

Regards,
Narasimhe

mohamed-taman commented on 15 Apr 2019

> Added this to dependency and it worked for me.
>
> ```
>         <dependency>
>             <groupId>org.springframework.plugin</groupId>
>             <artifactId>spring-plugin-core</artifactId>
>             <version>1.2.0.RELEASE</version>
>         </dependency>
> ```

It works for me also, thanks

👍 14 👎 5 🎉 2

ebussieres mentioned this issue on 25 Apr 2019

Spring 2.0.0.M1 springfox getPluginFor method did not exist #2968 Open

eceolin commented on 1 May 2019 • edited

I've fixed replacing some swagger classes

First of all, i created some configuration that gonna replace the Swagger Beans by the mine ones.

`public class OverrideSwaggerClasses {

```
@Autowired private TypeNameExtractorReplaced typeNameExtractorReplaced;

@Bean
@Primary
public DocumentationPluginsManager bean() {
    return new DocumentationPluginsManagerReplaced();
}

@Bean
@Primary
public TypeNameExtractor beanTypeName() {
    return typeNameExtractorReplaced;
}
```

}
`

After, i've to "implement" these bean...

`public class DocumentationPluginsManagerReplaced extends DocumentationPluginsManager {

```
@Autowired
@Qualifier("documentationPluginRegistry")
private PluginRegistry<DocumentationPlugin, DocumentationType> documentationPlugins;
@Autowired
@Qualifier("apiListingBuilderPluginRegistry")
private PluginRegistry<ApiListingBuilderPlugin, DocumentationType> apiListingPlugins;
@Autowired
@Qualifier("parameterBuilderPluginRegistry")
private PluginRegistry<ParameterBuilderPlugin, DocumentationType> parameterPlugins;
@Autowired
@Qualifier("expandedParameterBuilderPluginRegistry")
private PluginRegistry<ExpandedParameterBuilderPlugin, DocumentationType> parameterExpand
@Autowired
@Qualifier("operationBuilderPluginRegistry")
private PluginRegistry<OperationBuilderPlugin, DocumentationType> operationBuilderPlugins
@Autowired
@Qualifier("resourceGroupingStrategyRegistry")
private PluginRegistry<ResourceGroupingStrategy, DocumentationType> resourceGroupingStrat
@Autowired
@Qualifier("operationModelsProviderPluginRegistry")
private PluginRegistry<OperationModelsProviderPlugin, DocumentationType> operationModelsP
@Autowired
@Qualifier("defaultsProviderPluginRegistry")
private PluginRegistry<DefaultsProviderPlugin, DocumentationType> defaultsProviders;
@Autowired
@Qualifier("pathDecoratorRegistry")
private PluginRegistry<PathDecorator, DocumentationContext> pathDecorators;
@Autowired
@Qualifier("apiListingScannerPluginRegistry")
private PluginRegistry<ApiListingScannerPlugin, DocumentationType> apiListingScanners;

public Iterable<DocumentationPlugin> documentationPlugins() throws IllegalStateException
    List<DocumentationPlugin> plugins = documentationPlugins.getPlugins();
    DuplicateGroupsDetector.ensureNoDuplicateGroups(plugins);
    if (plugins.isEmpty()) {
        return singleton(defaultDocumentationPlugin());
    }
    return plugins;
}

public Parameter parameter(ParameterContext parameterContext) {
    for (ParameterBuilderPlugin each : parameterPlugins.getPluginsFor(parameterContext.ge
        each.apply(parameterContext);
    }
    return parameterContext.parameterBuilder().build();
}

public Parameter expandParameter(ParameterExpansionContext context) {
    for (ExpandedParameterBuilderPlugin each : parameterExpanderPlugins.getPluginsFor(con
        each.apply(context);
    }
    return context.getParameterBuilder().build();
}

public Operation operation(OperationContext operationContext) {
    for (OperationBuilderPlugin each : operationBuilderPlugins.getPluginsFor(operationCon
        each.apply(operationContext);
    }
    return operationContext.operationBuilder().build();
}


public ApiListing apiListing(ApiListingContext context) {
    for (ApiListingBuilderPlugin each : apiListingPlugins.getPluginsFor(context.getDocume
        each.apply(context);
    }
    return context.apiListingBuilder().build();
}

public Set<ModelContext> modelContexts(RequestMappingContext context) {
    DocumentationType documentationType = context.getDocumentationContext().getDocumentat
    for (OperationModelsProviderPlugin each : operationModelsProviders.getPluginsFor(docu
        each.apply(context);
    }
    return context.operationModelsBuilder().build();
}

public ResourceGroupingStrategy resourceGroupingStrategy(DocumentationType documentation
    return resourceGroupingStrategies.getPluginOrDefaultFor(documentationType, new Spring
}

private DocumentationPlugin defaultDocumentationPlugin() {
    return new Docket(DocumentationType.SWAGGER_2);
}

public DocumentationContextBuilder createContextBuilder(
        DocumentationType documentationType,
        DefaultsProviderPlugin defaultConfiguration) {
    return defaultsProviders.getPluginOrDefaultFor(documentationType, defaultConfiguratio
            .create(documentationType)
            .withResourceGroupingStrategy(resourceGroupingStrategy(documentationType));
}

public Function<String, String> decorator(final PathContext context) {
    return input -> {
        Iterable<Function<String, String>> decorators
                = pathDecorators.getPluginsFor(context.documentationContext()).stream()
                .map(each -> each.decorator(context)).collect(toList());
        String decorated = input;
        for (Function<String, String> decorator : decorators) {
            decorated = decorator.apply(decorated);
        }
        return decorated;
    };
}

public Collection<ApiDescription> additionalListings(final ApiListingScanningContext con
    final DocumentationType documentationType = context.getDocumentationContext().getDocu
    List<ApiDescription> additional = new ArrayList<>();
```

```
    for (ApiListingScannerPlugin each : apiListingScanners.getPluginsFor(documentationTyp
        additional.addAll(each.apply(context.getDocumentationContext()));
    }
    return additional;
}
```

}`

and

`public class DuplicateGroupsDetector {
private DuplicateGroupsDetector() {
throw new UnsupportedOperationException();
}

```
public static void ensureNoDuplicateGroups(List<DocumentationPlugin> allPlugins) throws I
    Map<String, List<DocumentationPlugin>> plugins = allPlugins.stream()
            .collect(groupingBy(
                    input -> ofNullable(input.getGroupName()).orElse("default"),
                    LinkedHashMap::new,
                    toList()));


    Iterable<String> duplicateGroups = plugins.entrySet().stream()
            .filter(input -> input.getValue().size() > 1)
            .map(Map.Entry::getKey)
            .collect(toList());
    if (StreamSupport.stream(duplicateGroups.spliterator(), false).count() > 0) {
        throw new IllegalStateException(String.format("Multiple Dockets with the same gro
                + "The following duplicate groups were discovered. %s", String.join(",",
    }
}
```

}
`

and

`public class TypeNameExtractorReplaced extends TypeNameExtractor {

```
private final TypeResolver typeResolver;
private final PluginRegistry<TypeNameProviderPlugin, DocumentationType> typeNameProviders
private final EnumTypeDeterminer enumTypeDeterminer;

public TypeNameExtractorReplaced(TypeResolver typeResolver, PluginRegistry<TypeNameProvid
    super(typeResolver, typeNameProviders, enumTypeDeterminer);
    this.typeResolver = typeResolver;
    this.typeNameProviders = typeNameProviders;
    this.enumTypeDeterminer = enumTypeDeterminer;
}

public String typeName(ModelContext context) {
    ResolvedType type = asResolved(context.getType());
    if (isContainerType(type)) {
        return containerType(type);
    }
    return innerTypeName(type, context);
}

private ResolvedType asResolved(Type type) {
    return typeResolver.resolve(type);
}

private String genericTypeName(ResolvedType resolvedType, ModelContext context) {
    Class<?> erasedType = resolvedType.getErasedType();
    GenericTypeNamingStrategy namingStrategy = context.getGenericNamingStrategy();
    ModelNameContext nameContext = new ModelNameContext(resolvedType.getErasedType(), con
    String simpleName = ofNullable(typeNameFor(erasedType)).orElse(typeName(nameContext))
    StringBuilder sb = new StringBuilder(String.format("%s%s", simpleName, namingStrateg
    boolean first = true;
    for (int index = 0; index < erasedType.getTypeParameters().length; index++) {
        ResolvedType typeParam = resolvedType.getTypeParameters().get(index);
        if (first) {
            sb.append(innerTypeName(typeParam, context));
            first = false;
        } else {
            sb.append(String.format("%s%s", namingStrategy.getTypeListDelimiter(),
                    innerTypeName(typeParam, context)));
        }
    }
    sb.append(namingStrategy.getCloseGeneric());
    return sb.toString();
}

private String innerTypeName(ResolvedType type, ModelContext context) {
    if (type.getTypeParameters().size() > 0 && type.getErasedType().getTypeParameters().l
        return genericTypeName(type, context);
    }
    return simpleTypeName(type, context);
}

private String simpleTypeName(ResolvedType type, ModelContext context) {
    Class<?> erasedType = type.getErasedType();
    if (type instanceof ResolvedPrimitiveType) {
        return typeNameFor(erasedType);
    } else if (enumTypeDeterminer.isEnum(erasedType)) {
        return "string";
    } else if (type instanceof ResolvedArrayType) {
        GenericTypeNamingStrategy namingStrategy = context.getGenericNamingStrategy();
        return String.format("Array%s%s%s", namingStrategy.getOpenGeneric(),
                simpleTypeName(type.getArrayElementType(), context), namingStrategy.getCl
    } else if (type instanceof ResolvedObjectType) {
        String typeName = typeNameFor(erasedType);
        if (typeName != null) {
            return typeName;
        }
    }
    return typeName(new ModelNameContext(type.getErasedType(), context.getDocumentationTy
}

private String typeName(ModelNameContext context) {
    TypeNameProviderPlugin selected =
            typeNameProviders.getPluginOrDefaultFor(context.getDocumentationType(), new D
    return selected.nameFor(context.getType());
}
```

}
`

It's just a hack to make it works...

My test is over https://github.com/duduceolin/webflux-tips/

👍 1

Purus commented on 13 May 2019 ···

Friendly ping for knowing about official fix for this issue..

👍 27

reddybdr commented on 4 Jun 2019 ···

> Added this to dependency and it worked for me.
>
> ```
> <dependency>
>     <groupId>org.springframework.plugin</groupId>
>     <artifactId>spring-plugin-core</artifactId>
>     <version>1.2.0.RELEASE</version>
> </dependency>
> ```

Worked for me as well

skuznets0v commented on 14 Jul 2019

> Added this to dependency and it worked for me.
>
> ```
> <dependency>
>     <groupId>org.springframework.plugin</groupId>
>     <artifactId>spring-plugin-core</artifactId>
>     <version>1.2.0.RELEASE</version>
> </dependency>
> ```

It does not work with HATEOAS module, because it needs spring-plugins-core 2.0+

👍 3

jahidakhtargit commented on 17 Jul 2019

Yes,

org.springframework.plugin
spring-plugin-core
1.2.0.RELEASE

Worked, but not with HATEOS module. What is the solution to run both HATEOS and Swagger together?

tksilicon commented on 23 Jul 2019

> Temporary solution:
> Create 2 primary beans as following:
> public class DocumentationPluginsManagerBootAdapter extends DocumentationPluginsManager {
> **@Autowired**
> **@qualifier**("resourceGroupingStrategyRegistry")
> private PluginRegistry<ResourceGroupingStrategy, DocumentationType>
> resourceGroupingStrategies;
>
> ```
> @Autowired
> @Qualifier("defaultsProviderPluginRegistry")
> private PluginRegistry<DefaultsProviderPlugin, DocumentationType> defaultsProviders;
>
> @Override
> public ResourceGroupingStrategy resourceGroupingStrategy(DocumentationType documenta
>         return resourceGroupingStrategies.getPluginOrDefaultFor(documentationType, n
> }
>
> @Override
> public DocumentationContextBuilder createContextBuilder(DocumentationType documentat
>                 DefaultConfiguration defaultConfiguration) {
>         return defaultsProviders.getPluginOrDefaultFor(documentationType, defaultCon
>                         .create(documentationType).withResourceGroupingStrategy(reso
> }
> ```
>
> }
>
> public class TypeNameExtractorBootAdapter extends TypeNameExtractor {
> private final TypeResolver typeResolver;
> private final PluginRegistry<TypeNameProviderPlugin, DocumentationType> typeNameProviders;
> private final EnumTypeDeterminer enumTypeDeterminer;
>
> ```
> public TypeNameExtractorBootAdapter(TypeResolver typeResolver,
>                 PluginRegistry<TypeNameProviderPlugin, DocumentationType> typeNamePr
>                 EnumTypeDeterminer enumTypeDeterminer) {
>         super(typeResolver, typeNameProviders, enumTypeDeterminer);
>         this.typeResolver = typeResolver;
>         this.typeNameProviders = typeNameProviders;
>         this.enumTypeDeterminer = enumTypeDeterminer;
> }
>
> public String typeName(ModelContext context) {
>         ResolvedType type = asResolved(context.getType());
>         if (isContainerType(type)) {
>                 return containerType(type);
>         }
>         return innerTypeName(type, context);
> }
>
> private ResolvedType asResolved(Type type) {
>         return typeResolver.resolve(type);
> }
>
> private String genericTypeName(ResolvedType resolvedType, ModelContext context) {
>         Class<?> erasedType = resolvedType.getErasedType();
>         GenericTypeNamingStrategy namingStrategy = context.getGenericNamingStrategy(
>         ModelNameContext nameContext = new ModelNameContext(resolvedType.getErasedTy
>                         context.getDocumentationType());
>         String simpleName = fromNullable(typeNameFor(erasedType)).or(typeName(nameCo
>         StringBuilder sb = new StringBuilder(String.format("%s%s", simpleName, namin
>         boolean first = true;
>         for (int index = 0; index < erasedType.getTypeParameters().length; index++)
>                 ResolvedType typeParam = resolvedType.getTypeParameters().get(index)
>                 if (first) {
>                         sb.append(innerTypeName(typeParam, context));
>                         first = false;
>                 } else {
>                         sb.append(String.format("%s%s", namingStrategy.getTypeListDe
>                                         innerTypeName(typeParam, context)));
>                 }
>         }
>         sb.append(namingStrategy.getCloseGeneric());
>         return sb.toString();
> }
>
> private String innerTypeName(ResolvedType type, ModelContext context) {
>         if (type.getTypeParameters().size() > 0 && type.getErasedType().getTypeParam
>                 return genericTypeName(type, context);
>         }
>         return simpleTypeName(type, context);
> }
>
> private String simpleTypeName(ResolvedType type, ModelContext context) {
>         Class<?> erasedType = type.getErasedType();
>         if (type instanceof ResolvedPrimitiveType) {
>                 return typeNameFor(erasedType);
>         } else if (enumTypeDeterminer.isEnum(erasedType)) {
>                 return "string";
>         } else if (type instanceof ResolvedArrayType) {
> ```

```
                GenericTypeNamingStrategy namingStrategy = context.getGenericNaming
                return String.format("Array%s%s", namingStrategy.getOpenGeneric(),
                                simpleTypeName(type.getArrayElementType(), context),
        } else if (type instanceof ResolvedObjectType) {
                String typeName = typeNameFor(erasedType);
                if (typeName != null) {
                        return typeName;
                }
        }
        return typeName(new ModelNameContext(type.getErasedType(), context.getDocume
}

private String typeName(ModelNameContext context) {
        TypeNameProviderPlugin selected = typeNameProviders.getPluginOrDefaultFor(co
                        new DefaultTypeNameProvider());
        return selected.nameFor(context.getType());
}
```

}

This fixed the problem. Working temporary solution. Mark your beans as **@primary**. Thanks.

👍 2

geekyouth commented on 29 Jul 2019

> Added this to dependency and it worked for me.
>
> ```
> <dependency>
>     <groupId>org.springframework.plugin</groupId>
>     <artifactId>spring-plugin-core</artifactId>
>     <version>1.2.0.RELEASE</version>
> </dependency>
> ```

Thanks ！！！❤👌😘

vishal12126 commented on 30 Jul 2019

Added this to Pom and it works:

org.springframework.plugin
spring-plugin-core
1.2.0.RELEASE

vrgkwd commented on 1 Aug 2019

> Added this to dependency and it worked for me.
>
> ```
> <dependency>
>     <groupId>org.springframework.plugin</groupId>
>     <artifactId>spring-plugin-core</artifactId>
>     <version>1.2.0.RELEASE</version>
> </dependency>
> ```
>
> Thanks ！！！❤

What about SpringBoot

HattoriHenzo commented on 2 Aug 2019 • edited

Unless I am missing something in my POM, it doesn't work for me and still have the same issue.

```
<project xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://maven.apache
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/x
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.2.0.BUILD-SNAPSHOT</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.practice</groupId>
    <artifactId>practice-bus-manager</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>bus-manager</name>
    <description>FullStack bus manager app</description>

    <properties>
        <java.version>1.8</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-actuator</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
        </dependency>
        <dependency>
            <groupId>nz.net.ultraq.thymeleaf</groupId>
            <artifactId>thymeleaf-layout-dialect</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.plugin</groupId>
            <artifactId>spring-plugin-core</artifactId>
            <version>1.2.0.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>io.jsonwebtoken</groupId>
            <artifactId>jjwt</artifactId>
            <version>0.9.1</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.modelmapper</groupId>
```

```
            <artifactId>modelmapper</artifactId>
            <version>2.3.2</version>
        </dependency>
        <dependency>
            <groupId>javax.persistence</groupId>
            <artifactId>persistence-api</artifactId>
            <version>1.0.2</version>
        </dependency>
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger2</artifactId>
            <version>2.9.2</version>
        </dependency>
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger-ui</artifactId>
            <version>2.9.2</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <!-- Test dependencies -->

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

Any idea please ?

tksilicon commented on 2 Aug 2019

Unless I am missing something in my POM, it doesn't work for me and still have the same issue.

```
<project xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://maven.a
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.o
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.2.0.BUILD-SNAPSHOT</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.practice</groupId>
    <artifactId>practice-bus-manager</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>bus-manager</name>
    <description>FullStack bus manager app</description>

    <properties>
        <java.version>1.8</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-actuator</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
        </dependency>
        <dependency>
            <groupId>nz.net.ultraq.thymeleaf</groupId>
            <artifactId>thymeleaf-layout-dialect</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.plugin</groupId>
            <artifactId>spring-plugin-core</artifactId>
            <version>1.2.0.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>io.jsonwebtoken</groupId>
            <artifactId>jjwt</artifactId>
            <version>0.9.1</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.modelmapper</groupId>
            <artifactId>modelmapper</artifactId>
            <version>2.3.2</version>
        </dependency>
        <dependency>
            <groupId>javax.persistence</groupId>
            <artifactId>persistence-api</artifactId>
            <version>1.0.2</version>
        </dependency>
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger2</artifactId>
            <version>2.9.2</version>
        </dependency>
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger-ui</artifactId>
            <version>2.9.2</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>
```

```
    <!-- Test dependencies -->

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

Any idea please ?

Implement the two classes @duduceolin mentioned. That is what my backend is using currently. My version is 2.2.0.M2.

vrgkwd commented on 2 Aug 2019

Hi,
While running the swagger program getting the following error,any idea how to solved this issue?

****
APPLICATION FAILED TO START
****

Description:

An attempt was made to call a method that does not exist. The attempt was made from the following location:

springfox.documentation.spring.web.plugins.DocumentationPluginsManager.createContextBuilder(DocumentationPluginsManager.java:152)

The following method did not exist:

org.springframework.plugin.core.PluginRegistry.getPluginFor(Ljava/lang/Object;Lorg/springframework/plugin/core/Plugin;)Lorg/springframework/plugin/core/Plugin;

The method's class, org.springframework.plugin.core.PluginRegistry, is available from the following locations:

jar:file:/C:/Users/vrgaikwad/.m2/repository/org/springframework/plugin/spring-plugin-core/2.0.0.M1/spring-plugin-core-2.0.0.M1.jar!/org/springframework/plugin/core/PluginRegistry.class

It was loaded from the following location:

file:/C:/Users/vrgaikwad/.m2/repository/org/springframework/plugin/spring-plugin-core/2.0.0.M1/spring-plugin-core-2.0.0.M1.jar

Action:

Correct the classpath of your application so that it contains a single, compatible version of org.springframework.plugin.core.PluginRegistry

...

Abhinav3414 commented on 6 Aug 2019 • edited

Is there any final solution which can be implemented other than adding below dependency in pom.xml (which is a temporary fix and breaks If you are using spring-plugin-core version 2. ??

```
		<dependency>
			<groupId>org.springframework.plugin</groupId>
			<artifactId>spring-plugin-core</artifactId>
			<version>1.2.0.RELEASE</version>
		</dependency>
```

Numbernick commented on 15 Aug 2019

I had the same issue. Also the code fixes are not working for me (The 3.0.0.SNAPSHOT is more a head).

So since I'm using **hateoas** as well my solution was to use the **1.2.0.RELEASE of the spring-plugin-core** and the **last version (0.25.2.RELEASE) of spring-hateoas**

dilipkrish added the **maintenance** label on 16 Aug 2019

dilipkrish commented on 16 Aug 2019 Member

Thanks to everyone who has reported. I've put this on the list to be fixed. Any PRs are also welcome

dilipkrish added this to the **3.0** milestone on 16 Aug 2019

thiagohmoreira mentioned this issue on 25 Aug 2019

springfox-swagger2 version 2.9.2 not compatible with springboot version 2.2.0.M4 #3052 Open

serenaponta mentioned this issue on 13 Sep 2019

Update rest-backend and rest-lib-utils to Spring 2.1.7 #167 Open

ghostd mentioned this issue on 20 Sep 2019

Compatibility with Spring Boot 2.2 #3124 Closed

nkonev added a commit to nkonev/blog that referenced this issue on 22 Sep 2019

`restore springfox` **`springfox/springfox#2932 (comment)`** ✕ 639a9c0

74 hidden items
Load more…

alienisty mentioned this issue on 12 Dec 2019

Issue 2932 #3210 Closed

aluckens commented on 12 Dec 2019

> Hi, I am running a Spring boot 2.2.0 with swagger version **2.9.2** and in swagger configuration file if I am using **@configuration** and **@EnableSwagger2**, getting the following error:
>
> APPLICATION FAILED TO START
>
> Description:
>
> An attempt was made to call a method that does not exist. The attempt was made from the following location:
>
> ```
> springfox.documentation.spring.web.plugins.DocumentationPluginsManager.createContext
> ```
>
> The following method did not exist:
>
> ```
> org.springframework.plugin.core.PluginRegistry.getPluginFor(Ljava/lang/Object;Lorg/s
> ```
>
> The method's class, org.springframework.plugin.core.PluginRegistry, is available from the following locations:
>
> ```
> jar:file:/Users/SystemName/.m2/repository/org/springframework/plugin/spring-plugin-c
> ```
>
> It was loaded from the following location:
>
> ```
> file:/Users/SystemName/.m2/repository/org/springframework/plugin/spring-plugin-core/
> ```
>
> Action:
>
> Correct the classpath of your application so that it contains a single, compatible version of org.springframework.plugin.core.PluginRegistry
>
> Any help would be appreciated.
> Thanks!

i fixed it by remove
org.springframework.boot:spring-boot-starter-data-rest
org.springframework.data:spring-data-rest-hal-browser Dependencies

👍 1

rougou commented on 12 Dec 2019

@aluckens Please note this project is dead. For the future, it will be better to switch to Springdoc when you get the chance.

shotinyuriy commented on 12 Dec 2019 • edited

> It's just a hack to make it works...
>
> My test is over https://github.com/duduceolin/webflux-tips/

That really worked for me. I was able to start the app. I was so easy to copy those files from github.

AndreasKl commented on 12 Dec 2019 Contributor

FYI: There is a ready to merge PR #3159 for this issue. Imho just contribute to this PR instead of creating new ones duplicating the same thing.

👍 1

NarahariP commented 28 days ago

Same issue I'm facing when i use combinedly Spring 2.2.0 HATEOAS and Swagger2.9.2. But those are working independently without any issues. Even it's working without dependency : spring-plugin-core.
If we configure spring-data-rest-hal-explorer along with Swagger2 its failing.
Can someone please guide how can i use both at a time any alternative?

Adriansun commented 28 days ago

> Same issue I'm facing when i use combinedly Spring 2.2.0 HATEOAS and Swagger2.9.2. But those are working independently without any issues. Even it's working without dependency : spring-plugin-core.
> If we configure spring-data-rest-hal-explorer along with Swagger2 its failing.
> Can someone please guide how can i use both at a time any alternative?

https://github.com/springdoc/springdoc-openapi

dilipkrish added a commit to beevolution/springfox that referenced this issue 20 days ago

`Merge branch 'master' into bugfix/springfox#2932-spring-boot-2.2.0` Verified `6f15b63`

dilipkrish added a commit that referenced this issue 20 days ago

`Merge branch 'bugfix/#2932-spring-boot-2.2.0' of https://github.com/b…` `c86fb6c`

dilipkrish added a commit that referenced this issue 20 days ago

`Merge branch 'beevolution-bugfix/#2932-spring-boot-2.2.0'` ✓ `ca57437`

dilipkrish commented 20 days ago Member

@rougou this project isn't dead, Im working to get back to supporting 3.0.

Closing this issue as its fixed via #3159

4

dilipkrish closed this 20 days ago

---

mrsarm commented 20 days ago

@dilipkrish , I see you closed the ticket pointing out another ticket that solve the issue, but if this is already fixed, why is not working? I couldn't find any clue in the ticket #3159 of how the issue was solved, could you summarize how set this library to work with Spring Boot 2.2+ ?

mrsarm commented 20 days ago

Update from my side: as suggested by **@aluckens** , removing the dependency `org.springframework.boot:spring-boot-starter-data-rest` that actually I wasn't using in my project, the problem was solved, so the issue is unresolved when you use it (at least using the latest version published of Springfox, now 2.9.2)

rougou commented 19 days ago

> **@rougou** this project isn't dead, Im working to get back to supporting 3.0.

My bad, I read in an earlier post that this wasn't being maintained anymore. Thanks for your efforts in getting it back on track.

@mrsarm

> I see you closed the ticket pointing out another ticket that solve the issue, but if this is already fixed, why is not working?

I don't see where he points to another ticket, just a PR that he merged to fix the issue. There is no release, yet, however.

SamNazar commented 19 days ago

Changing the version of `spring-plugin-core` from 1.2.0 to `2.0.0.RELEASE` solved this issue for us.

2

LeoSko commented 19 days ago

We faced same issue today when trying to build project, but the opposite way to OP's:

```
Description:

An attempt was made to call the method org.springframework.plugin.core.PluginRegistry.ge

    jar:file:/C:/project/lib/spring-plugin-core-1.2.0.RELEASE.jar!/org/springframework/p

It was loaded from the following location:

    file:/C:/projecta/lib/spring-plugin-core-1.2.0.RELEASE.jar


Action:

Correct the classpath of your application so that it contains a single, compatible versi
```

fixed by adding new dependency to match `2.0.0.RELEASE` instead of `1.2.0.RELEASE` to pom

```
        <dependency>
           <groupId>org.springframework.plugin</groupId>
           <artifactId>spring-plugin-core</artifactId>
           <version>2.0.0.RELEASE</version>
        </dependency>
```

3

vinisants commented 19 days ago

Same problem here since yesterday!

Fixed by adding:
compile group: 'org.springframework.plugin', name: 'spring-plugin-core', version: '2.0.0.RELEASE'

2

Bagurp referenced this issue in Bagurp/course-Api 19 days ago

`Update Spring Boot Parent version to 2.2.2` ✓ `4c53cfd`

paro87 commented 19 days ago

Started 3 microservices that use Swagger 3.0, with 20 seconds difference. The first and second started fine, the third one stopped with the above-mentioned problem. Restarted the first and second ones again, and they also stopped working. Solved with adding dependency spring-plugin-core 2.0.0.RELEASE. The strange thing is I used to start the whole project without any problem until now.

juan107 commented 18 days ago

> Started 3 microservices that use Swagger 3.0, with 20 seconds difference. The first and second started fine, the third one stopped with the above-mentioned problem. Restarted the first and second ones again, and they also stopped working. Solved with adding dependency spring-plugin-core 2.0.0.RELEASE. The strange thing is I used to start the whole project without any problem until now.

Same for me... until monday my microservice was working fine. But for some reason it stoped working. The workaround by adding the new dependency works for me, but i thinkg this is a temporary solution. Any one know what might be the root cause of this issue?

2

LeoSko commented 18 days ago

@juan107, this is simple - if you are using some snapshot maven repository, you need to accept the risk of breaking changes in snapshots. You never know when maintainers can change something in their development code branch, as well as you never know when repository will get updated with these breaking changes.
In our case we were using the following dependency from repo `http://oss.jfrog.org/artifactory/oss-snapshot-local/` :

```
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-spring-webflux</artifactId>
            <version>${springfox.version}</version>
            <scope>compile</scope>
        </dependency>
```

Johan-Lecuyer commented 17 days ago

> **@rougou** this project isn't dead, Im working to get back to supporting 3.0.
>
> Closing this issue as its fixed via #3159

Hello **@dilipkrish** ,
I have exactly the same problem as all the others except that I have tried all their solutions and nothing works ...
If your patch can correct this problem, when will the latest version go online?

Inesz referenced this issue in Inesz/StarWarsRestAPI 17 days ago

`bugfix problem: Failed to start bean 'documentationPluginsBootstrappe…` `5a8b714`

modikanisha commented 16 days ago

> Yes,
>
> org.springframework.plugin
> spring-plugin-core
> 1.2.0.RELEASE
> Worked, but not with HATEOS module. What is the solution to run both HATEOS and Swagger together?

@jahidakhtargit
Have you received any solution for this problem? My requirement is also same.
I need a solution. Please help me

wilkinsona mentioned this issue 16 days ago

gradle build issue #19601 Closed

radu-almasan commented 10 days ago

My approach to fixing this was do make use of the bean name resolution strategy: I added this:

```
    @Bean
    public PluginRegistry<LinkDiscoverer, MediaType> discoverers(
            OrderAwarePluginRegistry<LinkDiscoverer, MediaType> relProviderPluginRegistry
        return relProviderPluginRegistry;
    }
```

Splash34 commented 9 days ago

I got it working with the following setup:

#pom.xml

```
    <parent>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-parent</artifactId>
            <version>2.2.2.RELEASE</version>
            <relativePath />
    </parent>
```

```
      <dependencies>
            <dependency>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-starter-actuator</artifactId>
            </dependency>
            <dependency>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-starter-data-jpa</artifactId>
            </dependency>
            <dependency>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-starter-web</artifactId>
            </dependency>
            <dependency>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-starter-data-rest</artifactId>
            </dependency>
            <dependency>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-devtools</artifactId>
                    <scope>runtime</scope>
                    <optional>true</optional>
            </dependency>

            <!-- io.springfox setup  -->
            <dependency>
                    <groupId>io.springfox</groupId>
                    <artifactId>springfox-swagger-ui</artifactId>
                    <version>3.0.0-SNAPSHOT</version>
            </dependency>
            <dependency>
                    <groupId>io.springfox</groupId>
                    <artifactId>springfox-swagger2</artifactId>
                    <version>3.0.0-SNAPSHOT</version>
            </dependency>
            <dependency>
                    <groupId>io.springfox</groupId>
                    <artifactId>springfox-data-rest</artifactId>
                    <version>3.0.0-SNAPSHOT</version>
            </dependency>
    </dependencies>
```

```
    <repositories>
            <repository>
                    <id>jcenter-snapshots</id>
                    <name>jcenter</name>
```

```
                        <url>http://oss.jfrog.org/artifactory/oss-snapshot-local/</url>
                </repository>
        </repositories>
```

#SpringFoxConfig.java

```
@Configuration
@EnableSwagger2WebMvc
@Import(SpringDataRestConfiguration.class)
public class SpringFoxConfig {
        @Bean
        public Docket api() {
                return new Docket(DocumentationType.SWAGGER_2).select().apis(RequestHand
                                .paths(PathSelectors.any()).build();
        }
}
```

#URL:
http://localhost:8080/swagger-ui.html#/

I know it's just a snapshot version at the moment, but for my purpose it's totally fine for now.
Nevertheless, they have to release a new major version.

jaredstehler added a commit to wn-doolittle/springfox that referenced this issue 3 days ago

`pulling latest master (#1)` ··· Verified `b09f7cf`

jhonatasfender commented yesterday

I'm using version 2.4 and I had same problem.

What helped me to solve it, is in this link:

https://www.baeldung.com/swagger-2-documentation-for-spring-rest-api#springDataRest