How to make spring inject value into a static field

I know this may looks like a previously asked question but I'm facing a different problem here.

I have a utility class that has only static methods. I don't and I won't take an instance from it.

```
public class Utils{
    private static Properties dataBaseAttr;
    public static void methodA(){

    }

    public static void methodB(){

    }
}
```

Now I need Spring to fill dataBaseAttr with database attributes Properties.Spring config is:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:util="http://www.springframework.org/schema/util"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
    http://www.springframework.org/schema/util
http://www.springframework.org/schema/util/spring-util-3.0.xsd">

<util:properties id="dataBaseAttr"
        location="file:#{classPathVariable.path}/dataBaseAttr.properties" />
</beans>
```

I already done it in other beans but the problem here in this class (Utils) isn't a bean, And if I make it a bean nothing changes I still can't use the variable since the class will not be instantiated and variable always equals null.

java spring configuration app-config

edited Mar 19 '13 at 14:11

Duncan Jones
41.4k 11 98 154

asked Jul 4 '12 at 7:32

Osama Felfel
410 2 6 13

2 Answers

You have two possibilities:

1. non-static setter for static property/field;
2. using `org.springframework.beans.factory.config.MethodInvokingFactoryBean` to invoke a static setter.

In the first option you have a bean with a regular setter but instead setting an instance property you set the static property/field.

```
public void setTheProperty(Object value) {
    foo.bar.Class.STATIC_VALUE = value;
}
```

but in order to do this you need to have an instance of a bean that will expose this setter (its more like an *workaround*).

In the second case it would be done as follows:

```
<bean class="org.springframework.beans.factory.config.MethodInvokingFactoryBean">
    <property name="staticMethod" value="foo.bar.Class.setTheProperty"/>
    <property name="arguments">
        <list>
            <ref bean="theProperty"/>
        </list>
   </property>
</bean>
```

On you case you will add a new setter on the `Utils` class:

```
public static setDataBaseAttr(Properties p)
```

and in your context you will configure it with the approach exemplified above, more or less like:

```
<bean class="org.springframework.beans.factory.config.MethodInvokingFactoryBean">
    <property name="staticMethod" value="foo.bar.Utils.setDataBaseAttr"/>
    <property name="arguments">
        <list>
            <ref bean="dataBaseAttr"/>
        </list>
   </property>
</bean>
```

edited Jul 9 '15 at 9:57

Tomislav Nakic-Alfirevic
8,112 5 28 45

answered Jul 4 '12 at 7:36

Francisco Spaeth
17.6k 4 47 84

Thanks a lot, you really saved my day. – Osama Felfel Jul 4 '12 at 8:29

you are welcome! – Francisco Spaeth Jul 4 '12 at 8:46

I didn't try 1st solution since I didn't understand it well. I tried 2nd solution and it worked great. – Osama Felfel Jul 4 '12 at 8:58

4 Definitely prefer the second of these choices over the first. Setting static fields every time an instance is created is a really weird pattern. – Patrick May 30 '14 at 19:28

1 `staticMethod` is a property of `MethodInvokingFactoryBean` and is configured with the fully qualified static method name, wish will be invoked with the arguments provided to the property `argument` . – Francisco Spaeth Dec 23 '15 at 16:25

I've had a similar requirement: I needed to inject a Spring-managed repository bean into my `Person` entity class ("entity" as in "something with an identity", for example an JPA entity). A `Person` instance has friends, and for this `Person` instance to return its friends, it shall delegate to its repository and query for friends there.

```
@Entity
public class Person {
    private static PersonRepository personRepository;

    @Id
    @GeneratedValue
    private long id;

    public static void setPersonRepository(PersonRepository personRepository){
        this.personRepository = personRepository;
    }

    public Set<Person> getFriends(){
        return personRepository.getFriends(id);
    }

    ...
}
```

.

```
@Repository
public class PersonRepository {

    public Person get Person(long id) {
        // do database-related stuff
    }

    public Set<Person> getFriends(long id) {
        // do database-related stuff
    }

    ...
}
```

So how did I inject that `PersonRepository` singleton into the static field of the `Person` class?

I created a `@Configuration` , which gets picked up at *Spring ApplicationContext construction time*. This `@Configuration` gets injected with all those beans that I need to inject as static fields into other classes. Then with a `@PostConstruct` annotation, I catch a hook to do all static field injection logic.

```
@Configuration
public class StaticFieldInjectionConfiguration {

    @Inject
    private PersonRepository personRepository;

    @PostConstruct
    private void init() {
        Person.setPersonRepository(personRepository);
    }
}
```

answered Feb 5 '13 at 13:24

Abdull

11.4k 10 72 124

How is it possible you are able to access this from a static method ? I don't believe this is possible at all! Shouldn't it be Person.personRepository = personRepository ? – Jonas Geiregat May 8 '15 at 8:39

3 @JonasGeiregat it is possible as the method is static and it is accessing static variable – Anuj Acharya Nov 26 '15 at 19:19

3 I am wondering how someone reference a static variable through 'this' in a static context? – Kripz Apr 28 '16 at 19:02