WebSystique

learn together

Spring @PropertySource & @Value annotations example

Created on: August 19, 2014 | **Last updated on:** March 11, 2017 websystiqueadmin

In this post we will see how to read values from properties files using Spring `@PropertySource` & `@Value` annotations. We will also discuss about Spring `Environment` interface. We will see corresponding XML configuration as well for side-by-side comparison.

Spring @PropertySource annotations is mainly used to read from properties file using Spring's Environment interface. This annotation is in practice, placed on `@Configuration` classes. Spring @Value annotation can be used to specify expression on field or methods. Common use case is to specify the property from a .properties file along with default value. Let's see full example below.

Other interesting posts you may like

- Spring Boot+AngularJS+Spring Data+Hibernate+MySQL CRUD App
- Spring Boot REST API Tutorial
- Spring Boot WAR deployment example
- Spring Boot Introduction + Hello World Example

We use cookies to personalize content and ads, to provide the best browsing experience possible, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners. By continuing to use the site, you agree to the use of cookies. more information

Accept

- Spring 4 Caching Annotations Tutorial

WebSystiq

1 circ

+ 387

- Spring 4 Cache Tutorial with EhCache
- Spring MVC 4+AngularJS Example
- Spring 4 MVC+JPA2+Hibernate Many-to-many Example
- Spring 4 Email Template Library Example
- Spring 4 Email With Attachment Tutorial
- Spring 4 Email Integration Tutorial
- Spring MVC 4+JMS+ActiveMQ Integration Example
- Spring 4+JMS+ActiveMQ @JmsLister @EnableJms Example
- Spring 4+JMS+ActiveMQ Integration Example
- Spring MVC 4+Hibernate 4 Many-to-many JSP Example
- Spring MVC 4+Hibernate 4+MySQL+Maven integration example using annotations
- Spring MVC4 FileUpload-Download Hibernate+MySQL Example
- TestNG Mockito Integration Example Stubbing Void Methods
- Maven surefire plugin and TestNG Example
- Spring MVC 4 Form Validation and Resource Handling

Following technologies being used:

- Spring 4.0.6.RELEASE
- Maven 3
- JDK 1.6
- Eclipse JUNO Service Release 2

Project directory structure

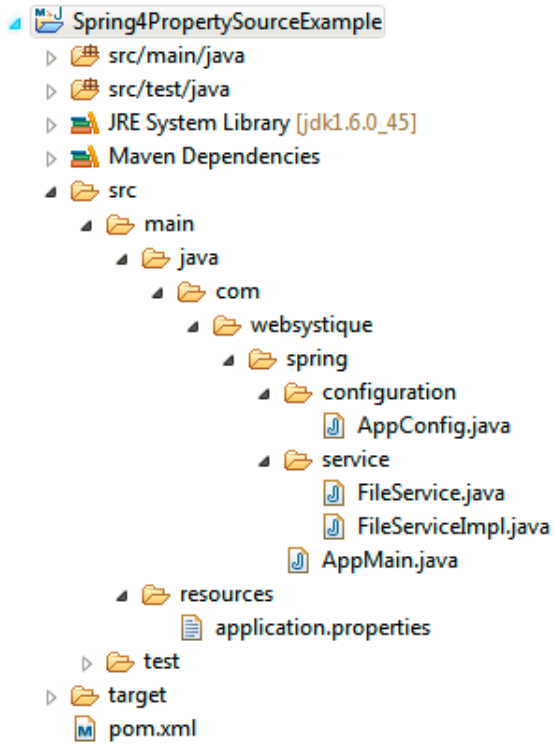
Following will be the final project directory structure for this example:

Recent Posts

- Spring Boot + AngularJS + Spring Data + JPA CRUD App Example
- Spring Boot Rest API Example
- Spring Boot WAR deployment example
- Spring Boot Introduction + hello world example
- Secure Spring REST API using OAuth2

We use cookies to personalize content and ads, to provide the best browsing experience possible, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners. By continuing to use the site, you agree to the use of cookies. more information

Accept



Let's add the content mentioned in above directory structure.

Step 1: Provide Spring dependencies in Maven pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns
        xsi:schemaLocation="http://maven.apache.org/POM/
        <modelVersion>4.0.0</modelVersion>

        <groupId>com.websystique.spring</groupId>
        <artifactId>Spring4PropertySourceExample</artifa
        <version>1.0.0</version>
        <packaging>jar</packaging>

        <name>Spring4PropertySourceExample</name>

        <properties>
                <springframework.version>4.0.6.RELEASE</
        </properties>

        <dependencies>
                <dependency>
                        <groupId>org.springframework</gr
                        <artifactId>spring-core</artifac
                        <groupId>org.springframework</gr
```

We use cookies to personalize content and ads, to provide the best browsing experience possible, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners. By continuing to use the site, you agree to the use of cookies. more information

Accept

```
                        <artifactId>spring-context</arti
                        <version>${springframework.versi
                </dependency>
        </dependencies>
        <build>
                <pluginManagement>
                        <plugins>
                                <plugin>
                                        <groupId>org.apa
                                        <artifactId>mave
                                        <version>3.2</ve
                                        <configuration>
                                                <source>
                                                <target>
                                        </configuration>
                                </plugin>
                        </plugins>
                </pluginManagement>
        </build>

</project>
```

Step 2: Create Spring Configuration Class

Spring configuration class are the ones annotated with `@Configuration`. These classes contains methods annotated with `@Bean`. These @Bean annotated methods generates beans managed by Spring container.

```
package com.websystique.spring.configuration;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentS
import org.springframework.context.annotation.Configurat
import org.springframework.context.annotation.PropertySo
i
@
@ComponentScan(basePackages = "com.websystique.spring")
```

We use cookies to personalize content and ads, to provide the best browsing experience possible, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners. By continuing to use the site, you agree to the use of cookies. more information

Accept

```
@PropertySource(value = { "classpath:application.propert
public class AppConfig {

        /*
         * PropertySourcesPlaceHolderConfigurer Bean onl
         * Remove this bean if you are not using @Value
         */
        @Bean
        public static PropertySourcesPlaceholderConfigur
                return new PropertySourcesPlaceholderCon
        }
}
```

@PropertySource(value = { “classpath:application.properties” }) annotation makes the properties available from named property file[s] (referred by value attribute) to Spring `Envirronment`. Environment interface provides getter methods to read the individual property in application.

Notice the `PropertySourcesPlaceholderConfigurer` bean method. This bean is required only for resolving ${…} placeholders in @Value annotations. In case you don’t use ${…} placeholders, you can remove this bean altogether.

Above Configuration can be expressed in XML based approach as follows (let’s name it app-config.xml):

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/bean
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-inst
        xmlns:context="http://www.springframework.org/sc
```

We use cookies to personalize content and ads, to provide the best browsing experience possible, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners. By continuing to use the site, you agree to the use of cookies. more information

Accept

```
        <bean class="org.springframework.context.support
            <property name="ignoreUnresolvablePlaceholde
            <property name="locations">
              <list>
                <value>classpath:application.properties<
              </list>
            </property>
          </bean>
</beans>
```

Step 3: Create Sample properties file

```
jdbc.driverClassName = com.mysql.jdbc.Driver
jdbc.url = jdbc:mysql://localhost:3306/websystique
jdbc.username = myuser
jdbc.password = mypassword
hibernate.dialect = org.hibernate.dialect.MySQLDialect
hibernate.show_sql = false
hibernate.format_sql = false
sourceLocation = /dev/input
```

We will read the properties from this file using above mentioned configuration in our sample service class.

Step 4: Create Sample service class

```
package com.websystique.spring.service;

public interface FileService {

        void readValues();
}
```

We use cookies to personalize content and ads, to provide the best browsing experience possible, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners. By continuing to use the site, you agree to the use of cookies. more information

Accept

```
p
import org.springframework.beans.factory.annotation.Auto
```

```
import org.springframework.beans.factory.annotation.Valu
import org.springframework.core.env.Environment;
import org.springframework.stereotype.Service;

@Service("fileService")
public class FileServiceImpl implements FileService {

        @Value("${sourceLocation:c:/temp/input}")
        private String source;

        @Value("${destinationLocation:c:/temp/output}")
        private String destination;

        @Autowired
        private Environment environment;

        public void readValues() {
                System.out.println("Getting property via
                                + environment.getPropert

                System.out.println("Source Location : "
                System.out.println("Destination Location

        }

}
```

First point to notice is Environment got auto-wired by Spring. Thanks to @PropertySoruce annotation , this Environment will get access to all the properties declared in specified .properties file. You can get the value of specif property using getProperty method. Several methods are defined in Environment interface.

Other interesting point here is `@Value` annotation. Format of value annotation is

```
@value("${key:default")
private String var;
```

Abov
.prop
found

We use cookies to personalize content and ads, to provide the best browsing experience possible, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners. By continuing to use the site, you agree to the use of cookies. more information

Accept

Note that above ${…} placeholder will only be resolved when we have registered `PropertySourcesPlaceholderConfigurer` bean (which we have already done above) else the @Value annotation will always assign default values to variable var.

Step 5: Create Main to run as Java Application

```
package com.websystique.spring;

import org.springframework.context.annotation.Annotation
import org.springframework.context.support.AbstractAppli

import com.websystique.spring.configuration.AppConfig;
import com.websystique.spring.service.FileService;

public class AppMain {

        public static void main(String args[]){
                AbstractApplicationContext  context = ne
                FileService service = (FileService) cont

                service.readValues();
                context.close();
        }

}
```

Run above program , you will see following output:

```
Getting property via Spring Environment :com.mysql.jdbc.
Source Location : /dev/input
Destination Location : c:/temp/output
```

Since destinationLocation property was not found in application.properties, it's got the default value.

That

We use cookies to personalize content and ads, to provide the best browsing experience possible, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners. By continuing to use the site, you agree to the use of cookies. more information
Accept

For ❱

```
AbstractApplicationContext  context = new Annota
```

with

```
AbstractApplicationContext context = new ClassPathXmlApp
```

in above main, no other changes. Run the program and you will see same output.

Download Source Code

Download Now!

References

- Spring framework
- @Value
- @PropertySource
- Spring Environment

websystiqueadmin

If you like tutorials on this site, why not take a step further and connect me on Facebook , Google Plus & Twitter as well? I would love to hear your thoughts on these articles, it will help me improve further our learning process.

If you appreciate the effort I have put in this learning site, help me improve the visibility of this site towards global audience by sharing and linking this site from within and beyond your network. You & your friends can always link my site from your site on www.websystique.com, and share the learning.

After all, we are here to learn together, aren't we?

Rel

We use cookies to personalize content and ads, to provide the best browsing experience possible, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners. By continuing to use the site, you agree to the use of cookies. more information

Accept

(Annotations+XML)

2. **Spring @Profile Guide**
3. **Spring Auto-detection autowire & Component-scanning Example With Annotations**
4. **Spring Dependency Injection Annotation Example, Beans Auto-wiring using @Autowired, @Qualifier & @Resource Annotations Configuration**

spring. permalink.

← Spring Auto-detection autowire & Component-scanning Example With Annotations

Spring @Profile Guide →

11 Comments websystique Login

Recommend Share Sort by Best

Join the discussion…

Faiyaz Md Abdul • 2 months ago
very nice tutorial ,learning a lot .
I compared this tutorial with mkyong.com , you have missed to handle FileNotFoundexception
@PropertySource(value="classpath:missing.properties", ignoreResourceNotFound=true)
• Reply • Share ›

Tapan Yawalkar • 7 months ago
Hi,

I am getting the following error while running the above application main class.
Exception in thread "main" org.springframework.beans.factory.NoSuchBeanDefinitionException: No bean named 'fileService' is defined
• Reply • Share ›

websystique Mod Tapan Yawalkar • 7 months ago
Hi Tapan, Please make sure that your service implementation class [FileServiceImpl] is annotated appropriately

We use cookies to personalize content and ads, to provide the best browsing experience possible, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners. By continuing to use the site, you agree to the use of cookies. more information

Accept

Thanks..issue resolved

• Reply • Share ›

Igor • 9 months ago

Why i have this?

Getting property via Spring Environment :com.mysql.jdbc.Driver
Source Location : ${sourceLocation:c:/temp/input}
Destination Location : ${destinationLocation:c:/temp/output}

• Reply • Share ›

websystique Mod ➚ Igor • 9 months ago

Hi Igor, With property configuraton, it will first look for the value of sourceLocation into properties, if it found, that value will be used, else default value [mentioned after :] will be used. If you are facing issue with them, it means your resolver is not configured correctly. Are you using Spring Boot?

• Reply • Share ›

Jesus Moscoso • 2 years ago

Thanks for the post. It's clear and i could follow pretty well but after trying it is not working to me.

I created a post in Stack overflow http://stackoverflow.com/qu... where i reference to this one.

Any ideas what could be wrong in my project.

• Reply • Share ›

websystique Mod ➚ Jesus Moscoso • 2 years ago

Hi,
Could you please mention the exact version of spring you are using?

• Reply • Share ›

Jesus Moscoso ➚ websystique • 2 years ago

Sure.

<spring.version>3.2.6.RELEASE</spring.version>

<dependency>
<groupid>org.springframework</groupid>
<artifactid>spring-web</artifactid>
<version>${spring.version}</version>
</dependency>
<dependency>
<groupid>org.springframework</groupid>
<artifactid>spring-webmvc</artifactid>
<version>${spring.version}</version>

We use cookies to personalize content and ads, to provide the best browsing experience possible, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners. By continuing to use the site, you agree to the use of cookies. more information

Accept

<version>${spring.version}</version>
</dependency>

</dependency>

• Reply • Share ›

websystique Mod ➚ Jesus Moscoso
• 2 years ago

Hi Jesus,

I tested this example with 3.2.6.RELEASE without any issue. Anyway, Is there a way you could share your minimal runnable code you are trying to run (through github for example)? That would help me to pinpoint particular issue you are getting. You can also use 'contact us' page of this site to send the details if you prefer.

• Reply • Share ›

Jesus Moscoso ➚ websystique • 2 years ago

Yes, you are right. I found there is something else.

I can execute this in @Controller classes and in @Services classes, but i also have a @Service class that implements UserDetailsService class.

Copyright © 2014-2016 WebSystique.com. All rights reserved.

We use cookies to personalize content and ads, to provide the best browsing experience possible, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners. By continuing to use the site, you agree to the use of cookies. more information

Accept