Injecting values to different types (simple, array, list, set, map, HAS-A property) of spring bean properties from properties file

It is recomanded to add the following dependency in pom.xml while working with

@ConfigurationProperties annotaiton to get META DATA (more info/more details) about spring bean propeties Simply we get suggesssions/hint box in applicaiton.properties towards user-defined spring bea properties.

<dependency>

file

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-configuration-processor</artifactId>

<optional>true</optional>

</dependency>

file

=>The allowed or recomanded special characters in keys of properties are

U can add this

through suggestion (optional to add)

Generate metadata for developers to offer contextual help and "code completion" when working with custom configuration keys (ex.application.properties/.yml files).

"_","-","["","]" .

=> while working with array, list,set type bean properties we must provide sequential index to elements #Set type property - <prefix>.<prop>[index]=value

emp.info.phoneNumbers[0]=99999999999

emp.info.phone-numbers[1]=88888888888

emp.info.phone-numbers[2]=77777777777

In spring bean class

private Set<Long> phoneNumbers;

ctrl+shift+c :: To enable or disable

one line comment any where

(In properties file or java file or xml file or jsp file or ...)

emp.info.phone-numbers[4]=667777777777 // gives error becoz indexing not sequential

3 is expected here

っ

Property: emp.info.phone-numbers[4]

Value: 667777777777

Origin: class path resource [application.properties] - 19:27

Reason: The elements [emp.info.phone-numbers[4]] were left unbound.

=>The "." symbol in each key of properties file represents one level or node or key or sub property that depends on how and where we are using them

example App

//Compa===

package com.nt.sbeans; import lombok.Data; @Data

public class Company { private Integer id; private String name; private String addrs; private String size;

}

note: no need to this

class as spring bean becoz

it is operated through main

class "Employee"

application.properties

=========

#Employee Info

# simple properties

# <prefix>.<prop>=<value>

org.nit.eno=101

org.nit.ename=rajesh

# array properties

==========

#<prefix>.<prop>[index]=<value>

org.nit.favColors[0]=red

org.nit.favColors[1]=green

org.nit.favColors[2]=blue

# (or)

#org.nit.favColors=red,green, blue

(It is inline syntax)

#Collection properties

# for List/Set -- <prefix>.<prop>[index]=<value>

# List

#org.nit.nickNames[0]=bunty

#org.nit.nickNames[1]=chunty

#org.nit.nick-names[2]=chotu

#(or)

org.nit.nick-names-bunty, chunty, chotu

# Set

#org.nit.phoneNumbers[0]=99999999

#org.nit.phoneNumbers[1]=88888888

//Employee.java package com.nt.sbeans;

import java.util.List;

```
import java.util.Map;
import java.util.Set;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.stereotype.Component;
import lombok.Data;
@Component("emp")
@Data
@ConfigurationProperties(prefix="org.nit")
public class Employee {
//simple properties
private Integer eno;
private String ename;
// array type
private String[] favColors;
// Collecton type
private List<String> nickNames;
private Set<Long> phoneNumbers;
private Map<String, Object> idDetails;
// HAS-A property
private Company company;
```

Client napp

```
package com.nt;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication; import
org.springframework.context.ApplicationContext;
import com.nt.sbeans.Employee;
@SpringBootApplication
public class BootProj06ConfigurationPropertiesOnArrayListSetHasAPropertiesApplication {
public static void main(String[] args) {
}//main
}//class
//get IOC container
ApplicationContext ctx-SpringApplication.run(BootProj06Configuration
PropertiesOnArrayListSetHasAPropertiesApplication.class, args); //get Employee obj ref
Employee emp=ctx.getBean("emp", Employee.class);
//display the object data
System.out.println(emp);
#org.nit.phone-numbers[2]=77777777
```

# (or)

org.nit.phone-numbers-9999999,888888,7777777,7777777

#Map

# for Map -- <prefix>.<prop>.<key>=<value>

org.nit.idDetails.aadharid=AA44777GG667 org.nit.idDetails.voterid=BA54777GT657
org.nit.idDetails.passportno=IND54546777 prop name keys

values

# <prefx>.<prop>.<subprop>=<value>

# HAS -A property

org.nit.company.id=14567

org.nit.company.name=HCL

org.nit.company.addrs=Blore

this will be ignored becoz

Set does not support duplicates

BootProj06-Configuration PropertiesOnArray-List-Set-HAS-AProperties [boot]

> Spring Elements

#src/main/java

com.nt

> BootProj06ConfigurationPropertiesOnArrayListSetHasAPropertiesApplication

>

com.nt.sbeans

Company.java

note:: inline syntax to set values

is given only for array,list,set type properties.. not for the map, HAS-A properties

> Employee.java

#src/main/resources

application.properties

>#src/test/java

> JRE System Library [JavaSE-17]

Maven Dependencies

>

>

src

To disable the banner of the spring boot app

In application.properties

spring.main.banner-mode=off

(or)

In main (-) mehod of main class

SpringApplication app=new SpringApplication (BootlocProj07MoreOnConfiguration PropertiesAnnotationApplication.class);

org.nit.company.size=100000 स T

HAS-A property sub property

> target

WHELP.md

mvnw

mvnw.cmd

Mpom.xml >RemoteSystems TempFiles

=>The appicaiton.properties file can contain only user-defined keys or only pre-defined keys or both (mix of both)

=> the application.properties can following categories of pref-defined keys

1. Core Properties 2. Cache Properties 3. Mail Properties4. JSON Properties 5. Data Properties 6. Transaction Properties 7. Data Migration Properties 8. Integration Properties 9. Web Properties 10. Templating Properties 11. Server Properties 12. Security Properties 13. RSocket Properties 14. Actuator Properties 15. Devtools properties 16. Testing Properties

properties.html

note:: Spring boot f/w internally uses snakeyml api(snakeyml-<ver>.jar) to read and process yml documents/files

app.setBannerMode (Banner.Mode.OFF);

//get IOC container

ApplicationContext ctx-app.run(args);

//get Employee class obj

Employee emp=ctx.getBean("emp", Employee.class);

System.out.println(emp);

//close the container

((ConfigurableApplicationContext) ctx).close();

YML/YAML

=>Yet Another Markup language (or)

=>YAMLing language (or)

=>Yaint Markup language (best)

=>spring f/w is not supporting yml. only spring boot is supporting yml

=>The file can have either .yml or yaml extension..

application.properties

emp.info.name=raja

emp.info.id=1001

emp.info.location=hyd

prefix is repeated

=>The biggest limitation of properites files

is the nodes/levels will be repeated

in multiple keys.. especially while working

https://docs.spring.io/spring-boot/docs/current/reference/html/application-

with common prefix concept, collections, HAS-A properties to suoport bulk Injection using @ConfigurationProperties.

application.yml

emp:

info:

here prefix is not repeated

name: raja

id: 1001 location: hyd

to

org.nit.favColors[0]=red org.nit.favColors[1]=green org.nit.favColors[2]=blue prefix is repeated.

conclusion: if the keys are having more prefer properties file

=>Spring boot App internally converts every yml content properties content before using the content

compare to properties file preparation

the yml file preparation takes lesss time with experience..

repeated nodes or levels then prefer yml files otherwise

=> To convert yml content to properites content and to parse,process yml documents, spring boot internally

Example App

//Customer.java

package com.nt.sbeans;

note: The application.properties file or application.yml file of the

be

src/main/resources folder will recognized automatically by IOC container during the spring boot app startup

import org.springframework.boot.context.properties.ConfigurationProperties; import org.springframework.stereotype.Component;

import lombok.Data;

@Component("cust")

@ConfigurationProperties(prefix="cust.info")

@Data

public class Customer {

private Integer custNo; private String custName;

application.yml

# customer information

cust:

info:

custNo: 101

custName: raja

custAddrs: hyd billAmt: 5678.55

note: yml file data can be injected to

spring bean properties either using @Value

or using @Configuration Properties

use snakeyml api

(snakeyml-<ver>.jar)

(will be added spring starter

project automatically)

How to configure custom banner for spring boot application startup process? Ans) create banner.txt using online having ur project title

https://springhow.com/spring-boot-banner-generator/ -->

select style ::standard ---> type text:: Open Fx --> see preview and

download the file (banner.txt)

=> place banner.txt file in src/main/resources folder

=> Run the application

```
/_\_.
||||'_\/_\__\ T|_ \\/ /
| |_| | |_) | ____/ | || | _| > < __|__||_||_| /_/\___\
|__|
```

2024-05-08T12:35:08.397+05:30 INFO 21296

[BootProj07-B 2024-05-08T12:35:08.400+05:30 INFO 21296 --- [BootProj07-B 2024-05-08T12:35:09.096+05:30 INFO 21296 - [BootProj07-B CompanyDetails(cid=1001, cname=rajesh, addrs=hyd, owners=

private String custAddrs;

private Double billAmt;

}

in

note: application.yml file placed src/main/resources folder will be recognized

by spring boot as part of bootstrap process.

note: if we place both application.properites and application.yml file having same keys and different values then the application.properties data overrides the application.yml file content.

not

=> if newly added things or modified things are reflecting while executing eclipse IDE app, then it recomanded to clean the Project/App once Project menu ---> clean ---->clean

ClientApp

===========

```
package com.nt;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication; import
org.springframework.context.ApplicationContext; import
org.springframework.context.ConfigurableApplicationContext;
import com.nt.sbeans.Customer;
@SpringBootApplication
public class BootProj07YmlApplication {
public static void main(String[] args) {
// get IOC container
ApplicationContext ctx-SpringApplication.run(BootProj07YmlApplication.class, args);
//get Customer obj ref
Customer customer-ctx.getBean("cust",Customer.class);
System.out.println(customer);
//close IOC container
((ConfigurableApplicationContext) ctx).close();
}
}
```

While writing yml files from propeties files

(a) same nodes/levels in the keys should not be reperated

(b) replace "." each node/level with ":" symbol and write new node in the

next line having proper indentation (minimum single space is required)_(recomanded to take 3 spaces)

(c) replace "=" symobl with ":" before placing value having minimum single space

(d) To place array/list/set elements use "-" (hyphen) symbol having single space

(e) Take map collection keys and "HAS-A" proerty sub keys as the new nodes/levels

(f) use # symbol for commenting..

Example application.yml

```
#Employees info
BootProj08-Complex-Yml [boot]
org:
> Spring Elements
nit:
#src/main/java
#for simple properties
eno: 1001
ename: rajesh
com.nt.sbeans
> Company.java
```

# for array property

> Employee.java

favColors:

- red

- blue

- green

# for List collection

nickNames:

- chinna

- munna

- kanna

# for Set collection

phoneNumbers:

- 99999999

88888888

-77777777

# for map collection

idDetails:

aadhar: 898989888

voter: 88854ADG

panNo: 455777AA

# for HAS-A property

company:

id: 89012

name: HCL

addrs: hyd

size: 300

//Company.java

package com.nt.sbeans;

com.nt

> BootProj06Configuration PropertiesOnArrayListSetHasAPropertiesApplication.java

import lombok.Data;

@Data

public class Company {

#src/main/resources

application.yml

>src/test/java

> JRE System Library [JavaSE-17]

> Maven Dependencies

> src

> target

WHELP.md

mvnw

mvnw.cmd

pom.xml

if we give different set of values to same array/list/set properties

using both indexed approach and inline approach then which values will be

taken as the final values?

private Integer id;

private String name;

private String addrs; private String size;

Ans) in properties file --> the inline syntax values will be taken as the final values

# array properties (inline approach)

org.nit.favColors=red,green,blue

#(or)

The inline syntax values will be taken as the final values

#<prefix>.<prop>[index]=<value>

org.nit.favColors[0]=red1

org.nit.favColors[1]=green1

indexed approch

org.nit.favColors[2]=blue1

In yml file-->

nick-names:

- chinna1

these values

- munna1

will be taken as the

- kanna1

→inline values

final values

nickNames: [chinna, munna, kanna]

Q) Can we place values in properties file/yml file in a single line for array/list/set collection type properties?
Ans) yes, possible (Technically this mechanism is called inline approach of passing values)

//Employee.java package com.nt.sbeans;

```
import java.util.List;
import java.util.Map;
import java.util.Set;
import org.springframework.boot.context.properties.ConfigurationProperties; import
org.springframework.stereotype.Component;
import lombok.Data;
@Component("emp")
@Data
@Configuration Properties(prefix="org.nit")
public class Employee {
}
//simple properties
private Integer eno;
private String ename;
// array type
private String[] favColors;
// Collecton type
private List<String> nickNames;
private Set<Long> phoneNumbers;
private Map<String, Object> idDetails; // HAS- A property
private Company company;
```

application.properties

```
emp.info.nick-names[0]=chinna
emp.info.nick-names[1]=kanna
emp.info.nick-names[2]=munna
```

is eqaul to

```
emp.info.nick-names-chinna,kanna,munna
```

In line fromatting

appplication.yml

```
emp: info:
nick-names:
- chinna
kanna
- munna
```

is equal to

```
emp: info:
nick-names: [chinna,kanna,munna]
```

In line formatting

Q) if we place both applicaiton.properties and application.yml having same keys and different values

can u tell me what happens?

Ans) The values given application.properties will override the values given in applicaiton.yml

Q) can we pass data to spring bean properties of spring boot app from both application.properites file

and application.yml file?

Ans) yes possible, the properties of spring bean can we get few values from properties file and few other values from yml file . if we place different values for the same keys in application.properties file and application.yml file then application.properties file value will be taken as the final value

application.properties

#DataSource cfg

spring.datasource.driver-class-name-oracle.jdbc.driver.Oracle Driver

spring.datasource.url=jdbc:oracle:thin:@localhost:1521:xe

spring.datasource.username=system

spring.datasource.password=manager

application.properties #DataSource cfg

spring.datasource.driver-class-name-oracle.jdbc.driver.Oracle Driver
spring.datasource.url=jdbc:oracle:thin:@localhost:1521:xe

spring.datasource.username=system

spring.datasource.password=manager

c3p0.minsize=10

c3p0.maxsize=1000

spring.datasource.type=oracle.ucp.jdbc.PoolDataSourceImpl

application.yml

application.yml

spring:

datasource:

spring:

driver-class-name: oracle.jdbc.driver.Oracle Driver

datasource:

url: jdbc:oracle:thin:@localhost:1521:xe

username: system

password: manager

applicaiton.properties

my.nit.name=raja my.nit.id=1001

my.nit.one.profile= SE my.nit.one.exp =20 my.nit.two.qlfy-Engineer

application.properties

emp.info.name= raja

```
emp.info.id=2001
emp.info.addrs=hyd
cust.info. name= rajesh
cust.info.jd=3001
cust.info.billAmt=300
```

application.yml

亣

```
my:
```

application.yml

```
nit:
name: raja
emp:
id: 1001
info:
one:
profile : SE exp: 20
two:
name: raja id: 2001 addrs: hyd
qlfy: engineer
cust:
info:
name:rajesh
id: 3001
bill-amt: 300
```

application.yml

```
spring:
ds:
driver: oracle.jdbc.driver.Oracledriver
url: jdbc:oracle:thin:@localhost:1521:xe
db:
username: raja
password: hyd
boot:
banner:
enable :OFF
```

application.properties

```
driver-class-name: oracle.jdbc.driver.Oracle Driver
```

c3p0:

url: jdbc:oracle:thin:@localhost:1521:xe

username: system

password: manager type: oracle.ucp.jdbc.PoolDataSourceImpl

minsize: 10

maxsize: 1000

spring.ds.driver-oracle.jdbc.driver.Oracle Driver

spring.ds.url=jdbc:oracle:thin:@localhost:1521:xe

spring.ds.db.username=raja

spring.ds.db.password=hyd

spring.boot.banner.enable=OFF

note: the nodes/level in the keys of properties file or yml file are not case-sensitive.

=>once we have properties file in any eclipse spirng/spring boot project, then it can be converted

C ▲ Not secure | https://mageddo.com/tools/yaml-converter

MAGEDDO

Yaml to properties / Properties to Yaml converter

<< to yaml to properties >>

Yaml

into yml easily ussing wizard supplied by STS plugin

right click on properties file ----> convert to .yaml file..

(or) use online tool http://mageddo.com/tools/yaml-converter

for properties file to yml file and reverse

menu:

```
dosa:
price: '100'
wada:
price: '60'
```

Properties

```
menu.dosa.price=100
menu.wada.price=60
menu.poha.price=40
menu.idly.price=50
```

convert >>

```
cust.addrs-hyd cust.name=raja
```

poha:

```
price: '40'
idly:
```

```
price: '50'
cust:
addrs: hyd
```

<< convert

```
name: raja
```

What is the diffrence b/w properties file and yml file? properties file

======

=====

(a) There is no specification providing rules

yml file

=======

(b) There is sepcification providing

and guidelines to develop properites files

it is just keys-values

in

(b) can be used only in java

(c) No way related to JSON format

rules and guidelines to develope

the yml file (www.yaml.org)

(b) can be used java,phyton, ruby,groovy and etc..

(c) super set of JSON (yml is created on the top of JSON)

(d) can be used in both spring f/w and spring boot f/w (d) can not used in spring f/w .. i.e supported only in spring boot f/w

(e) nodes/levels in the keys may have duplicates

(e) same nodes/levels will not repeated (no duplicates)

```
YAML Resources:
YAML Specifications:
- YAML 1.2:
- Revision 1.2.2
- Revision 1.2.1
```

# Oct 1, 2021 *New*

```
# Oct 1, 2009
- Revision 1.2.0
# Jul 21, 2009
YAML 1.1
YAML 1.0
```

JSON:: Java Script Object Notation

(Xml, JSON are the two global formats

to define data.. JSON is best compare to XML)

note: JSON, XML format data is language, technology, framework, platform independent data

(f) It is not hierarchal

data

(f) It is hierarchal data

(g) Custom properties file can be configured

(g) Custom yml file can be configured

in spring boot App directly by using

in spring boot App by using

Factory class is required

using spring.config.import key

@PropertySource and no PropertySource

(h) while working with profiles in spring/spring boot (h) we can place mutiple profiles in single import: entry is required

@PropertySource and by developing, specifying PropertySource Factory class

(or) spring:

config:

we can not place multiple profiles in single properites file

yml file having seperation with "---".

in the application.yml file

i) Spring or Spring boot App directly loads and reads

the properties content

(i) every yml file will be converted to properties file content before loading and reading

are less

(j) Use properties file when no.of are keys and

(j) Use yml file when no.of are keys are more and

the nodes/levels keys are not repeating

the nodes/levels in keys are repeating

(k) Gives bit extra performence

(1) takes more time for typing becoz of

(k) Gives bit less performence compare to properties file becoz

every yml file content should converted into properties file content internally before using..

repeated nodes in the keys

(1) takes less time becoz of the no repeated nodes in the keys

How to configure user-defined yml/yaml file to the spring boot application?

Ans) use spring:

config:

import: <filename> key in application.yml to specify the user-defined yml file name properties file name

(or)

use spring.config.import key in application.properties file to specify the user-defined yml/yaml file /properties file name

Example Application

myfile.yml (src/main/resources folder)

#custom yml file

org:

ntt:

cust:

cno: 1001

name: raja

addrs:

hyd

billAmt: 9000

use

(src/main/resources)

In application.yml

spring:

(or)

config:

import: myfile.yml

(src/main/resources)

application.properties

spring.config.import-myfile.yml

Customer.java (In com.nt.sbeans package of src/main/java folder)

Customer.java

package com.nt.sbeans;

import org.springframework.boot.context.properties.Configuration Properties;

import org.springframework.stereotype.Component;

import lombok.Data;

@Component("cust")

@ConfigurationProperties(prefix = "org.ntt.cust")

@Data

public class Customer {

private Integer cno;

private String name;

private String addrs;

private Float billAmt;

}

as

In yml or yaml files we can not the following list of words the nodes in the keys becoz they are reserved words

But this restriction is not there with properites file

There's a long list of reserved words with this behavior:

y|Y yes Yes YES|n|N|no|No|NO

|true|True|TRUE |false|False|FALSE

|on|On ON|off | Off|OFF

This applies to both YML and YAML files.

note:: Working with yml /yaml files in spring boot

Projects is more industry standard compare to properties file

we

Q) How can configure custom properties file in spring /spring boot Application

spring.config.import is an array type property using which

we can configure multiple yml files or properties files or mix of both

Ans) In Spring Apps using @PropertySource Annotation

In spring boot Apps using @PropertySource annotation or using

spring.config.import key of the application.properties file

or spring:

config:

In application.yml or application.properties file we can configure multiple properties files, yml files s at a time as the list of files as shown below

Best

spring:

import: key of the application.yml file

config:

import:

Example App

- Info.yml

============

- info1.properties

myfile.properties (src/main/resources folder)

#custom properties file

org.ntt.cust.no=1001

org.ntt.cust.name=raja

org.ntt.cust.addrs=hyd

org.ntt.cust.billAmt=9000

In applicaton.properites

spring.config.import-myfile.properties

(or)

In application.yml

spring:

config:

import: myflle.properties

Customer.java (In com.nt.sbeans package of src/main/java folder)

Customer.java

package com.nt.sbeans;

import org.springframework.boot.context.properties.ConfigurationProperties;

import org.springframework.stereotype.Component;

import lombok.Data;

@Component("cust")

@ConfigurationProperties(prefix = "org.ntt.cust")

@Data

public class Customer {

private Integer cno;

private String name;

private String addrs;

private Float billAmt;

}

=>We can link user-defined yml file(s) with application.properies/yml file

using spring:

=>We can link user-defined properties file(s) with application.properies/yml file

config:

(or) spring.config.import

import key

application.properties/yml file and user-defined properites/yml file which is linked with application.ytml/properties

file are having same key with different values, can u tell me what happens?

Ans) The value kept in the custom properties file or yml file will be taken as the final value