**Spring CustomEditorConfigurer Example using spring 4**

In this article I will discuss about Spring CustomEditor.

While coding often there is a requirement where you want Text value will automatically convert into a desired Datatype and Vice versa.

Think about Struts ActionForm Or Spring @PathParam , @Formparam

Where parameters value in request are String but in ActionForm you get it desired Type say Integer, Date etc.

@Path(“employee”)

@RequestParam({age})

Say doX(@Pathparam(“age”)Integer age){….}

In above code snippet age automatically convert String to integer. But How it is possible?

The Answer is java Bean’s PropertyEditor interface by implementing PropertyEditor you can convert a string to your Custom datatype and Vice versa . To do that you have to implement its two main methods

**public void setAsText(String text) and public String getAsText().**

In Spring We can do it through Spring Built-in Custom editor. Or you can create your Own Editor by extending Java Beans PropertyEditorSupport Class which implements PropertyEditor interface.

***Problem statement: In Spring Suppose I want when an employee gives his Address in String format that will automatic convert to Address Class and from Address it will Extract zip-code and State.***

To solve this problem, I need to implement a CustomEditor class which will take the String convert it in to Address Object and do the extract operation.

I assume “-“ is the separator when employee provide address in String format .

Apart from that I need to register this custom Editor in Spring application Context so before any bean initialization it should be initialized so that, for any bean where a conversion needed from String to Address Object, this Custom Editor will be invoked by Spring container.

We do this by creating a Custom Register which extends Spring PropertyEditorRegistrar class and registor our Custom Editor.

Step 1 : create Address Object and provide the logic to extract zip and state from string provided by Employee

**package** com.example.aware;

**public** **class** Address {

String adress;

String pin;

String state;

Address() {

}

Address(String text) {

init(text);

}

**private** **void** init(String text) {

String[] arr = text.split("-");

**this**.setAdress(arr[0]);

**this**.setPin(arr[1]);

**this**.setState(arr[2]);

}

**public** String getAdress() {

**return** adress;

}

**public** **void** setAdress(String adress) {

**this**.adress = adress;

}

**public** String getPin() {

**return** pin;

}

**public** **void** setPin(String pin) {

**this**.pin = pin;

}

**public** String getState() {

**return** state;

}

**public** **void** setState(String state) {

**this**.state = state;

}

@Override

**public** String toString() {

**return** "Address [adress=" + adress + ", pin=" + pin + ", state="

+ state + "]";

}

}

Step 2 : Create an Employee bean

**package** com.example.aware;

**import** java.util.Date;

**public** **class** Employee {

**private** Integer id;

**private** String firstName;

**private** String lastName;

**private** String designation;

**private** Address address;

//Setters and Getters

**public** Integer getId() {

**return** id;

}

**public** **void** setId(Integer id) {

**this**.id = id;

}

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** String getLastName() {

**return** lastName;

}

**public** **void** setLastName(String lastName) {

**this**.lastName = lastName;

}

**public** String getDesignation() {

**return** designation;

}

**public** **void** setDesignation(String designation) {

**this**.designation = designation;

}

**public** Address getAddress() {

**return** address;

}

**public** **void** setAddress(Address address) {

**this**.address = address;

}

@Override

**public** String toString() {

**return** "Employee [id=" + id + ", firstName=" + firstName

+ ", lastName=" + lastName + ", designation=" + designation

+ ", address=" + address + "]";

}

}

Observe address type is Address not String but from spring bean xml I pass String value that automatically converted to Address Object

Step 3: Create Custom editor which will convert String to Address Object

**package** com.example.aware;

**import** java.beans.PropertyEditorSupport;

**public** **class** CustomAddressEditor **extends** PropertyEditorSupport{

**public** **void** setAsText(String text) {

setValue(**new** Address(text.toUpperCase()));

}

}

Look in setAsText I pass Address Object and pass text value as constructor argument ,setvalue is method which inherit from PropertyEditorSupport, it set value to the bean , so here from String to Address conversion is done.

Step 4: Register this bean into Spring Container so it can initialize before any beam. It is act as Bean post processor

**package** com.example.aware;

**import** java.util.Date;

**import** org.springframework.beans.PropertyEditorRegistrar;

**import** org.springframework.beans.PropertyEditorRegistry;

**import** org.springframework.beans.propertyeditors.ClassEditor;

**import** org.springframework.beans.propertyeditors.StringArrayPropertyEditor;

**public** **class** CustomAddressEditorRegistrar **implements** PropertyEditorRegistrar {

@Override

**public** **void** registerCustomEditors(PropertyEditorRegistry registry) {

registry.registerCustomEditor(Address.**class**, **new** CustomAddressEditor());

}

}

Here I register CustomAddressEditor.

Step 5: Spring Bean Xml declaration

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-3.0.xsd"*>

<bean class=*"org.springframework.beans.factory.config.CustomEditorConfigurer"*>

<property name=*"propertyEditorRegistrars"*>

<list>

<bean class=*"com.example.aware.CustomAddressEditorRegistrar"*/>

</list>

</property>

</bean>

<!-- employee bean -->

<bean id=*"employee"* class=*"com.example.aware.Employee"*>

<property name=*"firstName"* value=*"Shamik"*/>

<property name=*"lastName"* value=*"Mitra"*/>

<property name=*"designation"* value=*"Tech Lead"*/>

<property name=*"address"* value=*"1,NiveditaLane-700003-westbengal"*/>

</bean>

</beans>

Step 6 : Test the application

package com.example.aware;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Main {

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext("configFiles/customEditor.xml");

Employee employee = (Employee) context.getBean("employee");

System.out.println(employee);

}

}

Output :

Employee [id=null, firstName=Shamik, lastName=Mitra, designation=Tech Lead, address=Address [adress=1,NIVEDITALANE, pin=700003, state=WESTBENGAL]]

Please note that in Spring there is many inbuilt CustomEditor try to use them. If your

Problem statement not matching with any of built in editor then only create your own custom Editor.