Spring 5

Framework means compressed model

Written on top of java+servlets+jsp

1 Light weight

2 Layered architecture include needed packages and ignore unnecessary

3 IOC- inversion of control pre populate objects that r injected to run at runtime

Dependency injection

4 Very imp-Spring MVC

* Model is business logic + POJO objects
* View is presentation layer
* Controller is servlets that decides everything

5 AOP-aspect oriented programming

Auditing, Loggings, security, declarative transactions these kind of support functions around business logic when to where to how to trigger

6 transactions

7 modular

8 no need to invent new wheel

Doesn’t suggest new ORM..uses the available ORM and suggests how to integrate into existing file.

9 Data access

1. Spring JDBC -> SQL
2. Spring ORM -> without SQL
3. Spring data -> without data Impl …..This is widely used with Spring boot framework that is written on top of spring

10 easy to integrate like struts(concentrates only on web app)

Layers:

Different layers:

Data- ~~JDBC ORM~~ OXM JMS (java msging system) ~~Transactions~~ || Web- ~~Web MVC~~, Servlets, ~~portlets,~~ struts

AOP - Aop aspects instrumentation

Core -Core Context ~~Beans~~ Expression lang { }

Testing

New…maven project..create simpleproj tick..next group id:com

Step 1: add jar files

<properties>

<spring.version>5.0.5.RELEASE</spring.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

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

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

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

</dependency>

</dependencies>

Step 2:

Main class…right click newclass

Setter getter tostring

Step 3:

Main/resources

Rightclick new spring bean config file

Give name(just name) next….context-4.3 bean-4.3 finish

An xml file will be created

Bean…means calls setter method..so setter based injection,.

Dependency

Class employee is dependant on address….so add reference in employee class

Private Address address;

Create address class and its attri..set get and tostring

Dependency injection:

1. Setter based injection

<property name="employeeId" value="1001"></property>

<property name="employeeName" value="Tom"></property>

<property name="salary" value="12000"></property>

<property name="age" value="23"></property>

<property name="address" ref="empAddress"></property>

<bean id=*"empAddress"* class=*"org.cap.demo.Address"*>

<property name=*"doorNo"* value=*"23/A"*></property>

<property name=*"stName"* value=*"West Car st"*></property>

<property name=*"city"* value=*"Chennai"*></property>

</bean>

1. Constructor based injection

Create a parametrised constructor for this…then pass values using constructor arg tag

<constructor-arg name=*"employeeId"* value=*"1001"*></constructor-arg>

<constructor-arg name=*"employeeName"* value=*"Tom"* ></constructor-arg>

<constructor-arg name=*"salary"* value=*"12000"*></constructor-arg>

<constructor-arg name=*"age"* value=*"23"*></constructor-arg>

<constructor-arg name=*"address"* ref=*"empAddress"*></constructor-arg>

Method based injection…..find

Scope supported

* Singleton
* Prototype
* Session
* Request
* Application
* Web socket

Application nd web socket r newly added

Prev had..global session used in portlet application

Session request application websocket r for webApp

Request is http scope..stateless

Session is login to logoff or until timeout

I will travel multiple pages thruout which data is needed

Singleton means..for that clas we can create only 1 object thruout the lifetime…by default all beans r singleton

When ever I request it needs to create object..then it is prototype

Scope=”prototype”

ApplicationContext context=**new** ClassPathXmlApplicationContext("myBeans.xml");

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

Employee employee1=(Employee) context.getBean("employee");

employee.setEmployeeName("Jack");

System.***out***.println(employee);

System.***out***.println(employee1);

Autowiring

We r manually referring the dependant obj using property tag or constructor tag…..autowiring will do it automatically

Spring contrainer automatically detects the depdasnt name

Diff types

1)no autowiring

2 byname

<bean id=*"employee"* class=*"org.cap.demo.Employee"* autowire=*"byName"*>

<!-- <property name="address" ref="empAddress"></property> -->

<bean id=*"address"* class=*"org.cap.demo.Address"*>

3by Type

4 constructor

5 auto detect

Byname no prob…takes the name

Bytype..if both address nd emopadd

Then ambuigyuity

<?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:context=*"http://www.springframework.org/schema/context"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.3.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.3.xsd"*>

<bean id=*"employee"* class=*"org.cap.demo.Employee"* autowire=*"constructor"*>

<!-- <property name="employeeId" value="1001"></property>

<property name="employeeName" value="Tom"></property>

<property name="salary" value="12000"></property>

<property name="age" value="23"></property> -->

<!-- <property name="address" ref="empAddress"></property> -->

<constructor-arg name=*"employeeId"* value=*"1001"*></constructor-arg>

<constructor-arg name=*"employeeName"* value=*"Tom"* ></constructor-arg>

<constructor-arg name=*"salary"* value=*"12000"*></constructor-arg>

<constructor-arg name=*"age"* value=*"23"*></constructor-arg>

<!-- <constructor-arg name="address" ref="empAddress"></constructor-arg> -->

</bean>

<bean id=*"empaddress"* class=*"org.cap.demo.Address"*>

<property name=*"doorNo"* value=*"23/A"*></property>

<property name=*"stName"* value=*"North Car st"*></property>

<property name=*"city"* value=*"Hyderbad"*></property>

</bean>

<bean id=*"address"* class=*"org.cap.demo.Address"*>

<property name=*"doorNo"* value=*"23/A"*></property>

<property name=*"stName"* value=*"North Car st"*></property>

<property name=*"city"* value=*"Chennai"*></property>

</bean>

</beans>

Chooses address…same name

Autowire had disadv

ambiguos

We cant autowire primitives

Overriding possibility

Constructor..take preference to same name…else othr name also exists

Bean life cycle

Init method’

Destroy method

AbstractApplicationContext context = **new** ClassPathXmlApplicationContext("myBeans.xml");

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

System.***out***.println(employee);

context.registerShutdownHook();

use abstract for getting the init and destroy method…

**public** **void** init\_method() {

System.***out***.println("Init method------------");

}

**public** **void** destroy\_method() {

System.***out***.println("Destroy method-----------");

}

<bean id=*"employee"* class=*"org.cap.demo.Employee"* init-method=*"init\_method"* destroy-method=*"destroy\_method"*>

<property name=*"employeeId"* value=*"1001"*></property>

<property name=*"employeeName"* value=*"Tom"*></property>

<property name=*"salary"* value=*"12000"*></property>

<property name=*"age"* value=*"23"*></property>

<property name=*"address"* ref=*"empAddress"*></property>

In between init nd constructor…constructor is invoked first

Internally object is destroyed but not shown to console..to show it.//we change applicationcontext to abstractapplicationcontext

Importing…………………learn…..doubt

Beanpostprocessor----interface

It has 2 methods:

Beforeinit\_method

Afterinit\_method

<import resource=*"demoBeans.xml"*/>

Diff types of context:

Context can b created in 3 ways

1.classpathxmlApplicationContext if xml is located under class path

2. FileSystemXmlApplicationContext if someotr path

ApplicationContext context=**new** FileSystemXmlApplicationContext("D:\\Users\\sushsh\\Desktop\\myBeans.xml");

If we delete bean file from classpath and paste else where..we can give path using fileSystem context and still access it.

3. WebApplicationContext purely for web container

Arraylist..readonly…random accesss

Linkedlist manipulations

17.07.18\

List program

**package** org.cap.demo;

**import** java.util.List;

**public** **class** CollectionDemo {

**private** List<String> names;

**private** List<Address> addresses;

**public** List<Address> getAddresses() {

**return** addresses;

}

**public** **void** setAddresses(List<Address> addresses) {

**this**.addresses = addresses;

}

**public** List<String> getNames() {

**return** names;

}

**public** **void** setNames(List<String> names) {

**this**.names = names;

}

}

<?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:context=*"http://www.springframework.org/schema/context"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.3.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.3.xsd"*>

<bean id=*"collDemo"* class=*"org.cap.demo.CollectionDemo"*>

<property name=*"names"*>

<list>

<value>Jack</value>

<value>Johny</value>

<value>Jensy</value>

<value>Sush</value>

</list>

</property>

<property name=*"addresses"*>

<list>

<ref bean=*"address"*/>

</list>

</property>

</bean>

<bean id=*"address"* class=*"org.cap.demo.Address"*>

<property name=*"doorNo"* value=*"23/A"*></property>

<property name=*"stName"* value=*"North Car st"*></property>

<property name=*"city"* value=*"Chennai"*></property>

</bean>

</beans>

**package** org.cap.demo;

**import** java.util.Collection;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** MainClass {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

ApplicationContext ctx=**new** ClassPathXmlApplicationContext("myBeans.xml");

CollectionDemo result=(CollectionDemo) ctx.getBean("collDemo");

System.***out***.println(result.getAddresses());

System.***out***.println(result.getNames());

}

}

**package** org.cap.demo;

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

**private** String doorNo;

**private** String stName;

**private** String city;

**public** Address() {

//System.out.println("address class no arg");

}

**public** String getDoorNo() {

**return** doorNo;

}

**public** **void** setDoorNo(String doorNo) {

**this**.doorNo = doorNo;

}

**public** String getStName() {

**return** stName;

}

**public** **void** setStName(String stName) {

**this**.stName = stName;

}

**public** String getCity() {

**return** city;

}

**public** **void** setCity(String city) {

**this**.city = city;

}

@Override

**public** String toString() {

**return** "Address [doorNo=" + doorNo + ", stName=" + stName + ", city=" + city + "]";

}

}

<property name=*"addresses"*>

<list>

<ref bean=*"address1"*/>

<ref bean=*"address2"*/>

<ref bean=*"address3"*/>

</list>

</property>

</bean>

<bean id=*"address1"* class=*"org.cap.demo.Address"*>

<property name=*"doorNo"* value=*"23/A"*></property>

<property name=*"stName"* value=*"North Car st"*></property>

<property name=*"city"* value=*"Chennai"*></property>

</bean>

<bean id=*"address2"* class=*"org.cap.demo.Address"*>

<property name=*"doorNo"* value=*"21"*></property>

<property name=*"stName"* value=*"Sherparpet street"*></property>

<property name=*"city"* value=*"Pune"*></property>

</bean>

<bean id=*"address3"* class=*"org.cap.demo.Address"*>

<property name=*"doorNo"* value=*"67/2"*></property>

<property name=*"stName"* value=*"Mylai"*></property>

<property name=*"city"* value=*"delhi"*></property>

</bean>

Address is reference class…so we give ref bean

Property file….same as map..but in this we save only string key value pairs..

Property Is class…that also under collections

For map inside we use entry

<property name=*"map"*>

<map>

<entry key=*"1001"* value=*"Tom"* />

<entry key=*"1002"* value=*"Jack"* />

<entry key=*"1001"* value=*"Kamal"* />

<entry key=*"1002"* value=*"Singh"* />

</map>

</property>

For property insdie we use

<props>

<prop key=*"12"*>Tom</prop>

<prop key=*"16"*>Dinesh</prop>

<prop key=*"17"*>sush</prop>

<prop key=*"17"*>deepu</prop>

<prop key=*"14"*>maddy</prop>

<prop key=*"10"*>rocky</prop>

</props>

Diff types of configurations:

1. Xml
2. Annotations(xml+anno) very imp – main reason is provides metadata partial xml
3. Java based config(anno+java) classes totally eliminate xml completely

@dem

Public @interface annotation\_name{

}

Whenever we use annotations,we have to use this line below.

<context:annotation-config></context:annotation-config>

<bean id=*"employee"* class=*"org.cap.emp.Employee"*>

<!-- <property name="employeeId" value="1001"></property> -->

<property name=*"employeeName"* value=*"Tom"*></property>

<property name=*"salary"* value=*"12000"*></property>

<property name=*"age"* value=*"23"*></property>

<property name=*"address"* ref=*"address2"*></property>

</bean>

<bean id=*"address2"* class=*"org.cap.emp.Address"*>

<property name=*"doorNo"* value=*"21"*></property>

<property name=*"stName"* value=*"Sherparpet street"*></property>

<property name=*"city"* value=*"Pune"*></property>

</bean>

}

@Autowired

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

**this**.address = address;

}

Autowire….no need to give address…automatic detects

@Autowired

@Qualifier("address1")

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

**this**.address = address;

}

@qualifier resolves ambiguous error if all names r diff from address

Setter based injection can be done on top of either fields..or on top of set method

1)required…means that attribute is needed..if not given,throws error

@Required

**public** **void** setEmployeeId(**int** employeeId) {

**this**.employeeId = employeeId;

}

2)autowired

@Autowired

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

**this**.address = address;

}

No newed to give in property the address reference…it will auto find,..

If no matching class..thn error..it finds matching attri class…else give qualifier..

3)qualifier

To avoid ambuguigy..

Address …and qualifier is address2…still qualifier will be taken

@Autowired

@Qualifier("address1")

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

**this**.address = address;

}

All these above operations we have done on set method..we can also we constructor dependency..that is same is done on top of constructor.

For constructor,we use constructor-arg instead of property tag

Qualifier cannot be used on top constructor..becz certain annotations cant be written on top of constructor nd one such is qualifier

Autowiring…no need to give constructors,comment off the address line

Bean life cycle method:

Init

Destroy

We usually do in bean method initmethod=”” destroy\_mtho=””…

Instead we will use annotation now

}

@PostConstruct

**public** **void** init() {

System.***out***.println("Init method......");

}

@PreDestroy

**public** **void** destroy() {

System.***out***.println("Destroy method......");

}

Add the below dependency to get postconstrct nd postdestyroy annotations

<!-- https://mvnrepository.com/artifact/javax.annotation/jsr250-api -->

<dependency>

<groupId>javax.annotation</groupId>

<artifactId>jsr250-api</artifactId>

<version>1.0</version>

</dependency>

Postconstrcut means..execute as soon as constructor is executed

Predestroy means.,..bfr bean is destroyed..execute this

AbstractApplicationContext context=**new** ClassPathXmlApplicationContext("demoBeans.xml");

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

System.***out***.println(emp);

context.registerShutdownHook();

use abstractapplicationcontext nd registershutdownhook…for displaying destroy

@import….for bean declaration

@resource---for file inclusion

Java based config

Create a java class…

Class has to b annoted with @configuration

Method has to b annoted with @bean

package org.cap.emp;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

import org.springframework.context.support.AbstractApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainClass {

public static void main(String[] args) {

// TODO Auto-generated method stub

AbstractApplicationContext context=new AnnotationConfigApplicationContext(JavaConfig.class);

Employee emp= context.getBean(Employee.class);

Employee emp1= context.getBean(Employee.class);

emp.setEmployeeName("Maratha");

System.out.println(emp);

System.out.println(emp1);

context.registerShutdownHook();

}

Instead of mybeans.xml we write JavaConfig.class..the new class that we created. Instead of beAN id we write the employee that has all attributes…

We create a new class instead of mybeans.xml

**package** org.cap.emp;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.context.annotation.Scope;

@Configuration

**public** **class** JavaConfig {

@Bean

@Scope("prototype")

**public** Employee getEmployee() {

Employee emp=**new** Employee();

emp.setEmployeeId(1001);

emp.setEmployeeName("Jack");

emp.setSalary(80000);

emp.setAge(23);

//emp.setAddress(address);

**return** emp;

}

@Bean

**public** Address getAddress() {

Address ad=**new** Address();

ad.setDoorNo("23");

ad.setStName("Omr road");

ad.setCity("chennai");

**return** ad;

}

}

package org.cap.emp;

import javax.annotation.PostConstruct;

import javax.annotation.PreDestroy;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.beans.factory.annotation.Qualifier;

import org.springframework.beans.factory.annotation.Required;

public class Employee {

private int employeeId;

private String employeeName;

private double salary;

private int age;

@Autowired

private Address address;

public Employee() {

System.out.println("employee no arg");

}

public Employee(int employeeId, String employeeName, double salary, int age, Address address) {

super();

this.employeeId = employeeId;

this.employeeName = employeeName;

this.salary = salary;

this.age = age;

this.address = address;

}

public int getEmployeeId() {

return employeeId;

}

//@Required

public void setEmployeeId(int employeeId) {

this.employeeId = employeeId;

}

public String getEmployeeName() {

return employeeName;

}

public void setEmployeeName(String employeeName) {

this.employeeName = employeeName;

}

public double getSalary() {

return salary;

}

public void setSalary(double salary) {

this.salary = salary;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

public Address getAddress() {

return address;

}

/\* @Autowired

@Qualifier("address1")\*/

public void setAddress(Address address) {

this.address = address;

}

@PostConstruct

public void init() {

System.out.println("Init method......");

}

@PreDestroy

public void destroy() {

System.out.println("Destroy method......");

}

@Override

public String toString() {

return "Employee [employeeId=" + employeeId + ", employeeName=" + employeeName + ", salary=" + salary + ", age="

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

}

}

Singleton objects…for dependant class to be singleton we go for method injection

Create another class….student.java and other confignamed myConfig.java

**package** org.cap.emp;

**public** **class** Student {

**private** **int** studId;

**private** String studName;

**public** **int** getStudId() {

**return** studId;

}

**public** **void** setStudId(**int** studId) {

**this**.studId = studId;

}

**public** String getStudName() {

**return** studName;

}

**public** **void** setStudName(String studName) {

**this**.studName = studName;

}

@Override

**public** String toString() {

**return** "Student [studId=" + studId + ", studName=" + studName + "]";

}

**public** Student(**int** studId, String studName) {

**super**();

**this**.studId = studId;

**this**.studName = studName;

}

**public** Student() {

}

}

**package** org.cap.emp;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

@Configuration

**public** **class** myConfig {

@Bean

**public** Student getStudbean() {

Student stud=**new** Student();

stud.setStudId(1001);

stud.setStudName("sush");

**return** stud;

}

}

package org.cap.emp;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

import org.springframework.context.support.AbstractApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainClass {

public static void main(String[] args) {

// TODO Auto-generated method stub

AbstractApplicationContext context=new AnnotationConfigApplicationContext(JavaConfig.class);

Employee emp= context.getBean(Employee.class);

Employee emp1= context.getBean(Employee.class);

emp.setEmployeeName("Maratha");

System.out.println(emp);

System.out.println(emp1);

context.registerShutdownHook();

Student student=context.getBean(Student.class);

System.out.println(student);

}

}

In mainclass ass the new student.class…still the file is JavaConfig.class…so Import for myConfig

**package** org.cap.emp;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.context.annotation.Import;

**import** org.springframework.context.annotation.Scope;

@Import({myConfig.**class**})

@Configuration

**public** **class** JavaConfig {

@Bean

@Scope("prototype")

**public** Employee getEmployee() {

Employee emp=**new** Employee();

emp.setEmployeeId(1001);

emp.setEmployeeName("Jack");

emp.setSalary(80000);

emp.setAge(23);

//emp.setAddress(address);

**return** emp;

}

@Bean

**public** Address getAddress() {

Address ad=**new** Address();

ad.setDoorNo("23");

ad.setStName("Omr road");

ad.setCity("chennai");

**return** ad;

}

}

Spring MVC

package org.cap.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.context.annotation.Configuration;

import org.springframework.web.servlet.config.annotation.EnableWebMvc;

import org.springframework.web.servlet.view.InternalResourceViewResolver;

import org.springframework.web.servlet.view.JstlView;

@Configuration

@EnableWebMvc

@ComponentScan({"org.cap"})

public class AppConfig {

@Bean

public InternalResourceViewResolver getViewresolver() {

InternalResourceViewResolver viewResolver=new InternalResourceViewResolver();

viewResolver.setViewClass(JstlView.class);

viewResolver.setPrefix("/WEB-INF/views/");

viewResolver.setSuffix(".jsp");

return viewResolver;

}

}

MyWebInitialiser no change

package org.cap.controller;

import org.cap.model.Pilot;

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.servlet.ModelAndView;

@Controller

public class MyController {

@RequestMapping("/hello")

public ModelAndView greetUser() {

String message="Hello, Good Morning";

return new ModelAndView("helloPage","greetings",message);

}

@RequestMapping("/validateLogin")

public String validateLogin(ModelMap map,@RequestParam("username") String username, @RequestParam("password") String password) {

if(username.equals("tom") && password.equals("tom123")) {

map.put("pilot", new Pilot());

return "pilotForm";

}

return "redirect:/";

}

/\*@RequestMapping(value="/savePilot",method=RequestMethod.POST)\*/

@PostMapping("/showPilot")

public String showPilotdetails(@ModelAttribute("pilot") Pilot pilot){

System.out.println(pilot);

return "showPilot";

}

}

Pilot.java

**package** org.cap.model;

**import** java.util.Date;

**public** **class** Pilot {

**private** **int** pilotId;

**private** String firstName;

**private** String lastName;

**private** Date dateOfBirth;

**private** Date dateOfJoin;

**private** Boolean isCertified;

**private** **double** salary;

**public** **int** getPilotId() {

**return** pilotId;

}

**public** **void** setPilotId(**int** pilotId) {

**this**.pilotId = pilotId;

}

**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** Date getDateOfBirth() {

**return** dateOfBirth;

}

**public** **void** setDateOfBirth(Date dateOfBirth) {

**this**.dateOfBirth = dateOfBirth;

}

**public** Date getDateOfJoin() {

**return** dateOfJoin;

}

**public** **void** setDateOfJoin(Date dateOfJoin) {

**this**.dateOfJoin = dateOfJoin;

}

**public** Boolean getIsCertified() {

**return** isCertified;

}

**public** **void** setIsCertified(Boolean isCertified) {

**this**.isCertified = isCertified;

}

**public** **double** getSalary() {

**return** salary;

}

**public** **void** setSalary(**double** salary) {

**this**.salary = salary;

}

**public** Pilot(**int** pilotId, String firstName, String lastName, Date dateOfBirth, Date dateOfJoin, Boolean isCertified,

**double** salary) {

**super**();

**this**.pilotId = pilotId;

**this**.firstName = firstName;

**this**.lastName = lastName;

**this**.dateOfBirth = dateOfBirth;

**this**.dateOfJoin = dateOfJoin;

**this**.isCertified = isCertified;

**this**.salary = salary;

}

**public** Pilot() {

}

@Override

**public** String toString() {

**return** "Pilot [pilotId=" + pilotId + ", firstName=" + firstName + ", lastName=" + lastName + ", dateOfBirth="

+ dateOfBirth + ", dateOfJoin=" + dateOfJoin + ", isCertified=" + isCertified + ", salary=" + salary

+ "]";

}

}

pilotForm.jsp

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<%@ taglib uri=*"http://www.springframework.org/tags/form"* prefix=*"form"* %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>Insert title here</title>

</head>

<body>

<fieldset style="width: *500px*; margin-left: *200px*;">

<legend align=*"top"*>Pilot Form</legend>

<form:form action=*"showPilot"* method=*"post"* modelAttribute=*"pilot"*>

<table align=*"center"*>

<tr>

<th colspan=*"3"*><h2>

<b>Pivot Details</b>

</h2></th>

</tr>

<tr>

<td>Pilot ID:</td>

<td><form:input type=*"text"* path=*"pilotId"* size=*"30"*/></td>

</tr>

<tr>

<td>First Name:</td>

<td><form:input type=*"text"* path=*"firstName"* size=*"30"*/></td>

</tr>

<tr>

<td>Last Name:</td>

<td><form:input type=*"text"* path=*"lastName"* size=*"30"*/></td>

</tr>

<tr>

<td>Date of Birth:</td>

<td><form:input type=*"text"* path=*"dateOfBirth"* /></td>

</tr>

<tr>

<td>Date of Joining:</td>

<td><form:input type=*"text"* path=*"dateOfJoin"* /></td>

</tr>

<tr>

<td>Is certified?:</td>

<td>

<form:radiobutton path=*"isCertified"* value=*"true"*/>yes

<form:radiobutton path=*"isCertified"* value=*"false"*/>no

</td>

</tr>

<tr>

<td>Salary:</td>

<td><form:input type=*"text"* path=*"salary"* size=*"30"*/></td>

</tr>

<tr>

<td colspan=*"3"*><input type=*"submit"* value=*"save"*></td>

</tr>

</table>

</form:form>

</fieldset>

</body>

</html>

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>Insert title here</title>

</head>

<body>

<h2>Pilot Details</h2>

Pilot Id: ${pilot.pilotId}<br>

Pilot Name: ${pilot.firstName} &nbsp;&nbsp;&nbsp; ${pilot.lastName}<br>

Date Of Birth: ${pilot.dateOfBirth}<br>

Date of Joining: ${pilot.dateOfJoin}<br>

Is Certified?: ${pilot.isCertified}<br>

Salary: ${pilot.salary}<br>

</body>

</html>

Validation:

3 diff ways

1. Java script validation
2. Ajax validation
3. Framework validation
4. Javascript is client side validation..all browsers have javascript engine.’

Javascript can b written in 3 ways.

Internal javascript

External javascript

Inline

*.errMessage*{

color:*red*;

font-size: *14px*;

font

}<!-- name="myform" -->

<!-- onsubmit="return validateForm()" -->

**package** org.cap.model;

**import** java.util.Date;

**import** javax.validation.constraints.Future;

**import** javax.validation.constraints.NotNull;

**import** javax.validation.constraints.Past;

**import** org.hibernate.validator.constraints.NotEmpty;

**import** org.hibernate.validator.constraints.Range;

**import** org.springframework.lang.NonNull;

**public** **class** Pilot {

**private** **int** pilotId;

@NotEmpty(message="\* First Name should not be empty")

**private** String firstName;

@NotEmpty(message="\* Last Name should not be empty")

**private** String lastName;

@Past(message="\* Please enter a past date")

**private** Date dateOfBirth;

@NotNull(message="Should enter a date")

@Future(message="\* Date has to be a future date")

**private** Date dateOfJoin;

**private** Boolean isCertified;

@Range(min=1000,max=100000, message="\* Salary must be between 1k to 1L")

**private** **double** salary;

**public** **int** getPilotId() {

**return** pilotId;

}

**public** **void** setPilotId(**int** pilotId) {

**this**.pilotId = pilotId;

}

**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** Date getDateOfBirth() {

**return** dateOfBirth;

}

**public** **void** setDateOfBirth(Date dateOfBirth) {

**this**.dateOfBirth = dateOfBirth;

}

**public** Date getDateOfJoin() {

**return** dateOfJoin;

}

**public** **void** setDateOfJoin(Date dateOfJoin) {

**this**.dateOfJoin = dateOfJoin;

}

**public** Boolean getIsCertified() {

**return** isCertified;

}

**public** **void** setIsCertified(Boolean isCertified) {

**this**.isCertified = isCertified;

}

**public** **double** getSalary() {

**return** salary;

}

**public** **void** setSalary(**double** salary) {

**this**.salary = salary;

}

**public** Pilot(**int** pilotId, String firstName, String lastName, Date dateOfBirth, Date dateOfJoin, Boolean isCertified,

**double** salary) {

**super**();

**this**.pilotId = pilotId;

**this**.firstName = firstName;

**this**.lastName = lastName;

**this**.dateOfBirth = dateOfBirth;

**this**.dateOfJoin = dateOfJoin;

**this**.isCertified = isCertified;

**this**.salary = salary;

}

**public** Pilot() {

}

@Override

**public** String toString() {

**return** "Pilot [pilotId=" + pilotId + ", firstName=" + firstName + ", lastName=" + lastName + ", dateOfBirth="

+ dateOfBirth + ", dateOfJoin=" + dateOfJoin + ", isCertified=" + isCertified + ", salary=" + salary

+ "]";

}

}

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<%@ taglib uri=*"http://www.springframework.org/tags/form"* prefix=*"form"* %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<style type=*"text/css"*>

*.errMessage*{

color:*red*;

font-size: *14px*;

font-style:*italic*;

font-weight:*bold*;

}

</style>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>Insert title here</title>

</head>

<body>

<fieldset style="width: *500px*; margin-left: *200px*;">

<legend align=*"top"*>Pilot Form</legend>

<form:form action=*"showPilot"* method=*"post"* modelAttribute=*"pilot"*>

<table align=*"center"*>

<tr>

<th colspan=*"3"*><h2>

<b>Pivot Details</b>

</h2></th>

</tr>

<tr>

<td>Pilot ID:</td>

<td><form:input type=*"text"* path=*"pilotId"* size=*"30"*/></td>

</tr>

<tr>

<td>First Name:</td>

<td><form:input type=*"text"* path=*"firstName"* size=*"30"*/></td>

<td>

<form:errors path=*"firstName"* cssClass=*"errMessage"*></form:errors>

</td>

</tr>

<tr>

<td>Last Name:</td>

<td><form:input type=*"text"* path=*"lastName"* size=*"30"*/></td>

<td>

<form:errors path=*"lastName"* cssClass=*"errMessage"*></form:errors>

</td>

</tr>

<tr>

<td>Date of Birth:</td>

<td><form:input type=*"text"* path=*"dateOfBirth"* /></td>

<td>

<form:errors path=*"dateOfBirth"* cssClass=*"errMessage"*></form:errors>

</td>

</tr>

<tr>

<td>Date of Joining:</td>

<td><form:input type=*"text"* path=*"dateOfJoin"* /></td>

<td>

<form:errors path=*"dateOfJoin"* cssClass=*"errMessage"*></form:errors>

</td>

</tr>

<tr>

<td>Is certified?:</td>

<td>

<form:radiobutton path=*"isCertified"* value=*"true"*/>yes

<form:radiobutton path=*"isCertified"* value=*"false"*/>no

</td>

</tr>

<tr>

<td>Salary:</td>

<td><form:input type=*"text"* path=*"salary"* size=*"30"*/></td>

<td>

<form:errors path=*"salary"* cssClass=*"errMessage"*></form:errors>

</td>

</tr>

<tr>

<td colspan=*"3"*><input type=*"submit"* value=*"save"*></td>

</tr>

</table>

</form:form>

</fieldset>

</body>

</html>

@Configuration

@EnableWebMvc

@ComponentScan({"org.cap"})

Normal config

package org.cap.config;

import static java.lang.annotation.ElementType.FIELD;

import static java.lang.annotation.ElementType.METHOD;

import static java.lang.annotation.RetentionPolicy.RUNTIME;

import java.lang.annotation.Retention;

import java.lang.annotation.Target;

@Retention(RUNTIME)

@Target({ FIELD, METHOD })

public @interface CourseCode {

int age();

}

==

**package** org.cap.config;

**import** java.lang.reflect.Method;

**import** org.cap.model.Pilot;

**public** **class** TestCustomAnnotation {

**public** **static** **void** main(String[] args) **throws** Exception {

// **TODO** Auto-generated method stub

Pilot p=**new** Pilot();

Method m=p.getClass().getMethod("sayHi");

CourseCode annot=m.getAnnotation(CourseCode.**class**);

System.***out***.println("Value is:"+annot.age());

}

}

==

/\*@CourseCode(age=21)\*/