Dependency Injection in Spring

This code is an example of how to do Dependency Injection in Spring. As Spring is MVC

* Model: SfgDiApplication.java
  + Get the myController bean and executes the method sayHello
* Controller:sfgdi.controllers
  + MyController –

**package** sam.springwork.sfgdi.controllers;

@Controller

**public** **class** MyController {

**public** String sayHello() {

System.***out***.println("Hello World!!!");

**return** "Hi Folks";

}

}

**package** sam.springwork.sfgdi;

MyController myController = (MyController) ctx.getBean("myController");

String greeting = myController.sayHello();

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

* In the above code an instance of MyController is created and the greeting is called. But this does not give flexibility if the greeting is to be done in multiple languages.

**Services:**

**package** sam.springwork.sfgdi.services;

**public** **interface** GreetingService {

String sayGreeting();

}

**package** sam.springwork.sfgdi.services;

**public** **class** GreetingServiceImpl **implements** GreetingService{

@Override

**public** String sayGreeting() {

**return** "Hello World";

}

}

* The interface class GreetingService with the sayGreeting() method allows for greeting in multiple languages. The GreetingServiceImpl does the greeting in English. This can now be integrated in a PropertyInjected, SetterInjected or ContructorInjected class.

**PropertyInjectedController:**

**public** **class** PropertyInjectedController {

//handle to GreetingService

**public** GreetingService greetingService;

**public** String getGreeting() {

**return** greetingService.sayGreeting();

}

}

**PropertyInjectedControllerTest:**

**class** PropertyInjectedControllerTest {

PropertyInjectedController controller;

@BeforeEach

**void** setUp() {

//Mimicking what the controller service would do

controller = **new** PropertyInjectedController();

controller.greetingService = **new** GreetingServiceImpl();

}

@Test

**void** getGreeting() {

System.***out***.println(controller.getGreeting());

}

}

* **In the test, create an instance of the PropertyInjectedController, then instantiate the greetingService, can be any language, and then test it with the getGreeting. The greetingService associated with the controller is called.**
* **In the PropertyInjectedController, the GreetingServince is not sent in as a parameter.**
* **The greetingService is not private**

**SetterInjectedController**

**public** **class** SetterInjectedController {

**private** GreetingService greetingService;

**public** **void** setGreetingService (GreetingService greetingService) {

**this**.greetingService = greetingService;

}

**public** String getGreeting() {

**return** greetingService.sayGreeting();

}

}

**class** SetterInjectedControllerTest {

SetterInjectedController controller;

@BeforeEach

**void** setUp() {

//Mimicking what Spring Framework would do

//Create an instance of the SetterInjectedController

controller = **new** SetterInjectedController();

//associate the greeting and then call the greeting

controller.setGreetingService(**new** GreetingServiceImpl());

}

@Test

**void** sayGreeting() {

System.***out***.println(controller.getGreeting());

}

}

* **In the SetterInjectedController, the GreetingServince is sent in as a parameter. This is a more flexible version than the PropertyInjectedController**
* **The greetingService is private**

**ConstructorInjectedController**

**public** **class** ConstructorInjectedController {

**private** **final** GreetingService greetingService;

**public** ConstructorInjectedController(GreetingService greetingService) {

**this**.greetingService = greetingService;

}

**public** String getGreeting() {

**return** greetingService.sayGreeting();

}

}

**class** ConstructorInjectedControllerTest {

ConstructorInjectedController controller;

@BeforeEach

**void** setUp() **throws** Exception {

//Mimicking what Spring Framework would do

controller = **new** ConstructorInjectedController(**new** GreetingServiceImpl());

}

@Test

**void** getGreeting() {

System.***out***.println(controller.getGreeting());

}

}

* **The greetingService is private and final**