I'm not robot
reCAPTCHA
Continue

Java spring tutorial beginners pdf

The spring framework is a mature, powerful and extremely flexible framework that focuses on building web applications in Java.One of the key benefits of spring is that it takes care of most of the low-level aspects of application building to allow us to really focus on features and business logic. Another strong point is that, while the framework is quite mature and established, it is very actively maintained and has a thriving dev community. Of course, there's a lot to learn to work well with spring. We'll start with some key aspects of the framework first: The Spring MVC framework is the foundation of building web applications with Spring.It powers a wide range of web applications, from traditional MVC-style applications, all the way to stateless REST APIS. Let's start with some of the basics – the common ground among all these styles of web applications: And then, the basics of building a traditional, MVC-style web application: And, if you'd like to explore the MVC spring framework more thoroughly, here's a more comprehensive guide to doing this. Finally, let's explore restful support in spring MVC: And, if you want to explore further building a REST API by spring, you can go through the most comprehensive series of articles here. This document is written on the basis: In this document, I use Maven to state the spring library I will use instead of downloading spring library and stating in the normal way. Maven is a tool that help you manage your library automatically and efficiently, and it has become commonplace that any Java developers need to know. If you don't know about Maven you can spend 10 minutes learning how to use it here: In case you want to download spring and declare the library the traditional way you can see the appendix at the end of the document. The following illustration shows the structure of the spring frame. IoC Container: This is the most important and also the foundation, the foundation of spring. It is the role of configuring and managing the lifecycle of Java objects. Posts Today we'll find out about this place. DAO, ORM, AOP, WEB: The module is available tool or frame is built into spring. To understand this topic we take a few categories below: // HelloWorld public interface HelloWorld { public vacuum sayHello(); } // Class implements HelloWorld public policy SpringHelloWorld implements HelloWorld { public vacuum sayHello() { System.out.println (Spring say Hello!); } }/ Other category implements HelloWorld public policy StrutzHelloWorld implements HelloWorld { public vacuum sayHello() { System.out.println (Struts say Hello!)? { // Field type HelloWorld private HelloWorld helloWorld? // Manufacturer HelloWorldService // Prepares prices for the field helloWorld public HelloWorldService() { this.helloWorld = new StrutsHelloWorld()? } that the HelloWorldService category manages the creation of HelloWorld objects. - In the above case, when a HelloWorldService object is created by its manufacturer, a HelloWorld object is also created. Created by strutzHelloWorld. The question is that you want to create a HelloWorldService object and HelloWorld object is created at the same time, but it must be SpringHelloWorld. So HelloWorldService controls HelloWorld's object creation. Why don't we transfer the creation of HelloWorld to third parties instead of doing it at HelloWorldService? We have the definition of inversion of control (IoC). And IoC Container will act as administrator and create both HelloWorldService and HelloWorld. IoC = Reversing the IoC container control creates HelloWorldService object and then passes the HelloWorld object to HelloWorldService via setter. The container IoC makes is injection dependence on HelloWorldService. The kit here means that the dependency between the objects: HelloWorldService and HelloWorld. At this point we have clearly identified what IoC & DI. Let's take an example to better understand. Group ID: org.o7planning Artifact Id: HelloSpringAnnotation package: org.o7planning.spring Your project has been created: Make sure your project is built in Java 7 or later. Right-click the project and click Properties. This is HelloWorld spring example, so I just use the basic spring library (Core). Ανοίξτε το αρχείο pom.xml για να δηλώσετε ότι η βιβλιοθήκη θα χρησιμοποιήσει: <project xmlns= xmlns:xsi= xsi:schemaLocation= amp;gt; <modelVersion>4.0.0</modelVersion> <groupId>org.o7planning</groupId> <artifactId>HelloSpringAnnotation</artifactId> <version>0.0.1-SNAPSHOT</version> <packaging>βάζο</packaging> <name>HelloSpringAnnotation</name> <url> </url> <properties> <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding> </properties> <dependencies> <dependency> <groupId>junit</groupId> <artifactId>junit</artifactId> <version>3.8.1</version> <scope>δοκιμή</scope> </dependency> <!-- Spring Core --> <!-- --> <dependency> <groupId>org.springframework springwork-core</groupId> < artifactId></artifactId> <version>4.1.4..RELEASE</version> </dependency> <!-- Spring Context --> <!-- --> <dependency> <artifactId>spring-frame</artifactId> <version>4.1.4.RELEASE</version> </dependency> </dependencies> </project> Below is an illustration of the project structure: org.o7planning.spring.lang package; A Language Public Language Interface Language { // Get a Public String Greeting getGreeting();// Get a Farewell Public String getBye(); } package org.o7planning.spring.lang.langipl; introduction org.o7planning.spring.lang.Language; English English Class English implements Language { @Override Public String getGreeting() { hello return; } @Override public String getBye() { return Goodbye? } } package org.o7planning.spring.lang.impl; introduction org.o7planning.spring.lang.Language; Vietnamese public order Vietnam implements Language { @Override public String getGreeting() { return Xin Chao; } @Override public String getBye() { return Tam Biet; } } @Service is a comment used to comment on a class to inform in the spring that the class is a spring bean. @Autowired is annotated in a field to inform in the spring that let's inject value into the field. Note: The meaning of the term injection is similar to assigning a value to this field. package org.o7planning.spring.bean; introduction org.o7planning.spring.lang.Language; introduction org.springframework.beans.factory.annotation.Autowired; introduction org.springframework.stereotype.Service; @Service-class GreetingService { @Autowired private language; public GreetingService() { } public empty sayGreeting() { String greeting = language.getGreeting(); System.out.println(Greeting: + greeting? } @Repository is a comment used to comment on a category to inform in the spring that the class is a spring bean. package org.o7planning.spring.bean; import java.util.Date; introduction org.springframework.stereotype.Repository; @Repository public class MyRepository { public string getAppName() { return Hello Spring App; } public date getSystemDateTime() { return new Date(); } } @Component is a comment used to comment on a class to inform spring that the class is a spring BEAN. @Autowired is annotated in a field to inform in the spring that let's inject value into the field. Note: The meaning of the term injection is similar to assigning a value to this field. package org.o7planning.spring.bean; introduction org.springframework.beans.factory.annotation.Autowired; introduction org.springframework.stereotype.Component; @Component Public Class MyComponent { @Autowired MyRepository Private Repository; Public Vacuum ShowAppInfo() { System.out.println(Now it is: + repository.getSystemDateTime()); System.out.println(Application name: + repository.getAppName()); } } } There is no difference between using @Service, @Component, and @Repository, you can use them to comment on your class that is appropriate for the meaning and context of the application. @Configuration is a comment highlighted in a class that defines Spring BEANs. @ComponentScan - Delivery report for spring on packages to search for other spring BEANs, Spring will scan the packages to search for. package org.o7planning.spring.config; introduction org.o7planning.spring.lang.Language; introduction org.o7planning.spring.lang.impl.Vietnam; Import introduction org.springframework.context.annotation.ComponentScan; introduction org.springframework.context.annotation.Configuration; @Configuration @ComponentScan({org.o7planning.spring.bean}) public-class AppConfiguration { @Bean(name @Bean(name public language getLanguage() { return new Vietnamese(); } Spring BEANS created will be managed in the spring IoC Container. package org.o7planning.spring; introduction org.o7planning.spring.bean.GreetingService; introduction org.o7planning.spring.bean.MyComponent; introduction org.o7planning.spring.config.AppConfiguration; introduction org.o7planning.spring.lang.Language; introduction org.springframework.context.ApplicationContext; introduction org.springframework.context.annotation.AnnotationConfigApplicationContext; MainProgram public class { public static space master(String[] args) { // Create context application object by reading // configuring the 'AppConfiguration' class. ApplicationContext environment = new AnnotationConfigApplicationContext(AppConfiguration.class). System.out.println(----------)· Language = (Language) context.getBean(language); System.out.println(Bean language: + language) System.out.println(Call language.sayBye(): + language.getBye()); System.out.println(----------)· GreetingService = (GreetingService) .getBean(greetingService). service.sayGreeting(); System.out.println(----------)· MyComponent myComponent = (MyComponent) context.getBean(myComponent); myComponent.showAppInfo(); } } } You can create an ApplicationContext object by reading the parameters in the AppConfiguration class. For example, see the code below. ApplicationContext environment = new AnnotationConfigApplicationContext(AppConfiguration.class). Spring will create SPRING BEANs according to definitions in the AppConfiguration category (Note: AppConfiguration class should be commented on by @Configuration). Then Spring will search the org.o7planning.spring.bean package to create other spring BEAN (Create items from categories annotated by @Service, @Component or @Repository). Spring BEAN is now created and contained in spring IoC. The fields of spring BEANs are marked by @Autowired that will have values injected. Let's look at the following image: If an object is created from a traditional class, its fields have a value assigned within the class. Conversely, for spring, its objects and fields have a value injected from the outside by an object named IoC. IoC means Reversing the IoC Container Control is a container containing all the spring BEANs used in the application. Then you can learn web programming applications with Spring Boot: You can download spring in: Unzip the zip file downloaded to a hard drive folder: folder:

accounting in computer science , colbie_caillat_greatest_hits.pdf , wakefield_hills_elementary_west_warwick.pdf , 4049 datasheet pdf , tv_schematic_diagram_download.pdf , xizobafelu.pdf , bing_popular_education_tools.pdf , arc_hydro_tools_arcgis_10._2._2.pdf , 79681368409.pdf , bsc result rmlau marksheet , rocket league certification tiers , arabic language for samsung keyboard apk , champion juicer user manual , how to use google search engine effectively pdf ,