* **Spring之java配置原理以及加载容器示例**
* Spring的配置类：即用了@Configuration、@Bean的java类。

@Configuration修饰类：相当于传统的xml文件

@Bean修饰方法：相当于传统的xml文件中的bean标签。它的返回值类型即是可以被注入的类对象。

* 手工加载“配置类”并初始化容器的写法

现在如果要加载这个配置类，可以用

AnnotationConfigApplicationContext context = new AnnotationConfigApplicationContext(配置类名.class);

历史上xml时代的写法

ApplicationContext ac = new ClassPathXmlApplicationContext("applicationContext.xml");

ApplicationContext ac = new ClassPathXmlApplicationContext(new String[]{"applicationContext.xml","dao.xml"});

（关于ClassPathXmlApplicationContext可见<http://www.blogjava.net/xcp/archive/2011/06/22/352821.html>）

* web项目运行时初始化容器的过程

在web项目中，我们当然不想写以上这样的语句，实际上框架就已经为我们做好了这一切。当请求到来时，一系列的自动加载都已经搭建好了。

现在的比如springboot项目运行时，我们不需要写这种语句，而是项目启动时SpringApplication类就会从/META-INF/spring.factories文件中读取到要加载哪些配置类，便把容器初始化好了。这些配置类的名字通常是xxxAutoConfiguration（比如RedisAutoConfiguration）。通过IDE追踪代码可以知道xxxAutoConfiguration一般又调用了一些属性类（比如RedisProperties）来配合完成属性注入。

这些属性类的@ConfigurationProperties注解指定了属性全称的前缀（比如@ConfigurationProperties(prefix = "spring.redis")）。如果我们要修改取值，则就在全局属性文件application.properties或application.yml中书写，能覆盖掉默认值。

你可能会担心由于各种流行组件本身也在发展，比如Redis组件自身发展而版本不同，它的属性设置会否有变化。这不必担心，因为springboot本就有版本号，它自行实施了一套各组件的版本搭配法，一般不会出问题。

历史上，我们也并未写这种语句，web.xml文件中写好容器初始化所要使用的xml文件就好了，它们的加载加载是用spring的监听器<listener-class>org.springframework.web.context.ContextLoaderListener来完成的。（另外，对于mvc功能，在web.xml中配置DispatcherServlet时就指定了mvc功能用的xml。一旦客户端请求发来，DispatcherServlet能成功创建出controller及注入它的@Autowire成员（包扫描或xml<bean>标签注入））。

* **示例代码**

这个例子很说明问题

1、maven只添入了spring-webmvc依赖，因依赖的传递性，实际上它使得引入了很多的jar包。

2、context的关闭看起来有close()和destroy()两个都可

3、对比“UserService对象的获取” 和 “UserDAO对象的获取”

|  |
| --- |
| “UserService对象的获取”是靠它处于扫描路径中且它标记了@Service，  “UserDAO对象的获取”是靠在配置类SomeConfig中有写了一个返回UserDAO对象的方法getUserDAO()且此方法被标记了@Bean。  当然如果在配置类中也写个返回UserService对象的方法也行。 |

这是项目文件，它的代码同时也写在下方表格的。

|  |  |
| --- | --- |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <groupId>com.qq</groupId>  <artifactId>testConfiguration</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>war</packaging>  <dependencies>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-webmvc</artifactId>  <version>4.3.7.RELEASE</version>  </dependency>  </dependencies>  </project> |  |
| **package** com.qq.model;  **import** java.util.List;  **import** org.springframework.context.annotation.AnnotationConfigApplicationContext;  **public** **class** Main {  **public** **static** **void** main(String arg[]) {  // 实例化容器  AnnotationConfigApplicationContext context = **new** AnnotationConfigApplicationContext(SomeConfig.**class**);  // 获取Bean对象  UserService us = context.getBean(UserService.**class**);  List<User> users = us.query();  **for** (User user : users) {  System.***out***.println(user.getName() + " , " + user.getPassword());  }  context.destroy();  context.close();  }  } |  |
| **package** com.qq.model;  **import** org.springframework.context.annotation.Bean;  **import** org.springframework.context.annotation.ComponentScan;  **import** org.springframework.context.annotation.Configuration;  @ComponentScan(basePackages="com.qq.model")  @Configuration  **public** **class** SomeConfig {    @Bean  **public** UserDAO getUserDAO(){  **return** **new** UserDAO();  }    /\* @Bean  public UserService getUserService(){  return new UserService();  }\*/  } |  |
| **package** com.qq.model;  **import** java.util.List;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.stereotype.Service;  @Service  **public** **class** UserService {  @Autowired  **private** UserDAO ud;    **public** List<User> query(){  **return** ud.query();  }  } |  |
| **package** com.qq.model;  **import** java.util.ArrayList;  **import** java.util.List;  **public** **class** UserDAO {  **public** List<User> query(){    List<User> users = **new** ArrayList<User>();  **for**(**int** i=0;i<5;i++){  User u = **new** User();  u.setName("name\_"+ i);  u.setPassword("password\_"+i);  users.add(u);  }  **return** users;  }  } |  |
| **package** com.qq.model;  **public** **class** User {  **private** String name;  **private** String password;  **public** String getName() {  **return** name;  }  **public** **void** setName(String name) {  **this**.name = name;  }  **public** String getPassword() {  **return** password;  }  **public** **void** setPassword(String password) {  **this**.password = password;  }  } |  |
|  |  |