康康的Spring初识

徐振康 整理

目录

[第一讲 全面阐释Spring及其各项功能 3](#_Toc442345068)

[1.1 Spring是什么 3](#_Toc442345069)

[1.2 IOC 控制反转 3](#_Toc442345070)

[1.3 依赖注入(Dependency Injection) 3](#_Toc442345071)

[1.4 为何要使用Spring 3](#_Toc442345072)

[1.5 使用Spring的好处 4](#_Toc442345073)

[1.5.1 如果使用Spring, 我们就不再需要手工控制事务 4](#_Toc442345074)

[1.5.2 如果使用spring, 我们也不需要处理复杂的事务传播行为 5](#_Toc442345075)

[1.6 轻量级与重量级概念的划分 6](#_Toc442345076)

[第二讲 搭建与测试Spring的开发环境 6](#_Toc442345077)

[2.1 使用Spring需要的jar 6](#_Toc442345078)

[2.2 spring的配置文件模版 7](#_Toc442345079)

[2.3 编写spring配置文件时，不能出现帮助信息的解决办法 7](#_Toc442345080)

[2.4 实例化spring容器 7](#_Toc442345081)

[2.5 从spring容器中得到bean 7](#_Toc442345082)

[第三讲 编码剖析Spring管理Bean的原理 8](#_Toc442345083)

[使用dom4j读取spring配置文件 8](#_Toc442345084)

[第四讲 Spring的三种实例化Bean的方式 8](#_Toc442345085)

[4.1 三种实例化bean的方式 8](#_Toc442345086)

[4.1.1 使用类构造器实例化 8](#_Toc442345087)

[4.1.2 使用静态工厂方法实例化 9](#_Toc442345088)

[4.1.3 使用实例工厂方法实例化: 9](#_Toc442345089)

[第五讲 配置Spring管理的bean的作用域 9](#_Toc442345090)

[5.1 Bean的作用域 9](#_Toc442345091)

# 第一讲 全面阐释Spring及其各项功能

## 1.1 Spring是什么

Spring是一个开源的控制反转(Inversion of Control ,IoC)和面向切面(AOP)的容器框架.它的主要目得是简化企业开发.

## 1.2 IOC 控制反转

|  |
| --- |
| public class PersonServiceBean {  private PersonDao personDao = new PersonDaoBean();    public void save(Person person){  personDao.save(person);  }  } |

PersonDaoBean 是在**应用内部**创建及维护的。所谓控制反转就是**应用本身不负责**依赖对象的创建及维护，依赖对象的创建及维护是由外部容器负责的。这样控制权就由应用转移到了外部容器，控制权的转移就是所谓反转。

## 1.3 依赖注入(Dependency Injection)

当我们把依赖对象交给外部容器负责创建，那么PersonServiceBean 类可以改成如下：

|  |
| --- |
| public class PersonServiceBean {  private PersonDao personDao ;  //通过构造器参数，让容器把创建好的依赖对象注入进PersonServiceBean，当然也可以使用setter方法进行注入。  public PersonServiceBean(PersonDao personDao){  this.personDao=personDao;  }  public void save(Person person){  personDao.save(person);  }  } |

所谓**依赖注入**就是指：**在运行期，由外部容器动态地将依赖对象注入到组件中。**

## 1.4 为何要使用Spring

至少在我看来，在项目中引入spring立即可以带来下面的好处

（1）降低组件之间的耦合度,实现软件各层之间的解耦。

（2）可以使用容器提供的众多服务，如：事务管理服务、消息服务等等。当我们使用容器管理事务时，开发人员就不再需要手工控制事务.也不需处理复杂的事务传播。

Controller

Service

DAO

（3）容器提供单例模式支持，开发人员不再需要自己编写实现代码。

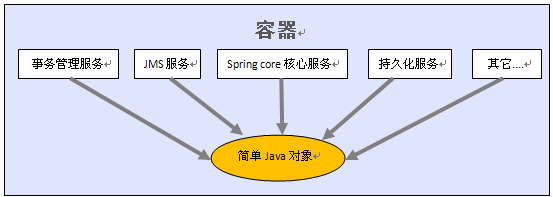
（4）容器提供了AOP技术，利用它很容易实现如权限拦截、运行期监控等功能。

（5）容器提供的众多辅作类，使用这些类能够加快应用的开发，如： JdbcTemplate、 HibernateTemplate。

（6）Spring对于主流的应用框架提供了集成支持，如：集成Hibernate、JPA、Struts等，这样更便于应用的开发。

## 1.5 使用Spring的好处

当使用spring时，我们可以使用容器提供的众多服务



### 1.5.1 如果使用Spring, 我们就不再需要手工控制事务

Hibernate的事务操作:

|  |
| --- |
| public void save(){  Session session = sessionFactory.getCurrentSession();  **~~session.beginTransaction();~~**  Info info = new Info("传智播客");  info.setContent("国内实力最强的java培训机构");  session.save(info );  **~~session.getTransaction().commit();~~**  } |

JDBC的事务操作:

|  |
| --- |
| Connection conn = null;  try {  .......  **~~conn.setAutoCommit(false);~~**  Statement stmt = conn.createStatement();  stmt.executeUpdate("update person where name='叶天'");  **~~conn.commit();~~**  .....  } **~~catch (Exception e) {~~**  **~~conn.rollback(); } finally{conn.close();}~~** |

### 1.5.2 如果使用spring, 我们也不需要处理复杂的事务传播行为

|  |
| --- |
| public void **payment**(){  Bean1.update();//更新金额  Bean2.save();//记录操作日志  } |

如果我们不使用Spring，针对下面这两种业务需求，我们该如何做？

第1种可能的业务需求：要求Bean1.update()和Bean2.save()在同一个事务中执行。

第2种可能的业务需求：要求不管Bean1.update() 的事务是否成功，都需要记录操作日志。

|  |
| --- |
| public class Bean1 {  public void update(){//注意：下面省略了一些代码  Connection conn = null;  conn.setAutoCommit(false);  Statement.executeUpdate(“update account set amount=? where id=?");  }  }  public class Bean2 {  public void save(){//注意：下面省略了一些代码  Connection conn = null;  conn.setAutoCommit(false);  Statement.executeUpdate(“insert into Log (content) values (?)");  }  } |

使用Spring，我们只需要通过声明式的事务属性配置就可以轻松地实现这两种业务需求

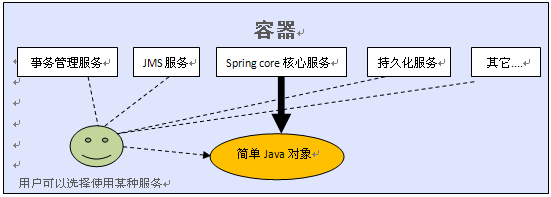
1.要求Bean1.update()和Bean2.save()的在同一个事务中执行

2.要求不管Bean1.update() 的事务是否成功，都需要记录日志。

|  |
| --- |
| @Transactional(propagation=Propagation.Required)  public void payment(){  Bean1.update();//更新金额  Bean2.save();//记录日志  }  public class Bean1 {  @Transactional(propagation=Propagation.Required)  public void update(){  executeUpdate(“update account set amount=? where id=?");  }  }  public class Bean2 {  @Transactional(propagation=Propagation.RequiresNew)  public void save(){  executeUpdate(“insert into Log (content) values (?)");  }  } |

## 1.6 轻量级与重量级概念的划分

经常会有同学问到spring属于轻量级框架,还是重量框架?其实划分一个应用是否属于轻量级还是重量级,主要看它使用了多少服务.使用的服务越多,容器要为普通java对象做的工作就越多,必然会影响到应用的发布时间或者是运行性能.



对于spring容器，它提供了很多服务，但这些服务并不是默认为应用打开的，应用需要某种服务，还需要指明使用该服务，如果应用使用的服务很少，如:只使用了spring核心服务，那么我们可以认为此时应用属于轻量级的，如果应用使用了spring提供的大部分服务，这时应用就属于重量级。目前EJB容器就因为它默认为应用提供了EJB规范中所有的功能，所以它属于重量级。

# 第二讲 搭建与测试Spring的开发环境

## 2.1 使用Spring需要的jar

**到http://www.springsource.org/download下载spring，然后进行解压缩，在解压目录中找到下面jar文件，拷贝到类路径下**

dist\spring.jar

lib\jakarta-commons\commons-logging.jar

如果使用了切面编程(AOP),还需要下列jar文件

lib/aspectj/aspectjweaver.jar和aspectjrt.jar

lib/cglib/cglib-nodep-2.1\_3.jar

如果使用了JSR-250中的注解,如@Resource/@PostConstruct/@PreDestroy,还需要下列jar文件

lib\j2ee\common-annotations.jar

## 2.2 spring的配置文件模版

|  |
| --- |
| **<?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-2.5.xsd">**  **.....**  **</beans>** |

该配置模版可以从spring的参考手册或spring的例子中得到。配置文件的取名可以任意，文件可以存放在任何目录下，但考虑到通用性，一般放在类路径下。

## 2.3 编写spring配置文件时，不能出现帮助信息的解决办法

由于spring的schema文件位于网络上，如果机器不能连接到网络，那么在编写配置信息时候就无法出现提示信息，解决方法有两种：

1.让机器上网，eclipse会自动从网络上下载schema文件并缓存在硬盘上。

2.手动添加schema文件,方法如下：

windwos->preferences->myeclipse->files and editors->xml->xmlcatalog

点"add",在出现的窗口中的Key Type中选择URI,在location中选"File system",然后在spring解压目录的dist/resources目录中选择spring-beans-2.5.xsd,回到设置窗口的时候不要急着关闭窗口,应把窗口中的Key Type改为Schema location,Key改为http://www.springframework.org/schema/beans/spring-beans-2.5.xsd

## 2.4 实例化spring容器

实例化Spring容器常用的两种方式：

方法一:

在类路径下寻找配置文件来实例化容器

ApplicationContext *ctx =* **new ClassPathXmlApplicationContext(new String[]{"beans.xml"});**

方法二:

在文件系统路径下寻找配置文件来实例化容器

ApplicationContext *ctx =* **new FileSystemXmlApplicationContext(new String[]{“d:\\beans.xml“});**

**Spring的配置文件可以指定多个，可以通过String数组传入。**

## 2.5 从spring容器中得到bean

当spring容器启动后，因为spring容器可以管理bean对象的创建，销毁等生命周期，所以我们只需从容器直接获取Bean对象就行，而不用编写一句代码来创建bean对象。从容器获取bean对象的代码如下：

|  |
| --- |
| ApplicationContext ctx = new ClassPathXmlApplicationContext(“beans.xml”);  **OrderService service = (OrderService)ctx.getBean("personService");** |

# 第三讲 编码剖析Spring管理Bean的原理

## 使用dom4j读取spring配置文件

|  |
| --- |
| public class ItcastClassPathXmlApplicationContext {  private List<BeanDefinition> beanDefines = new ArrayList<BeanDefinition>();  public ItcastApplicationContext(String filename){  init(filename);  }  private void init(String filename){  SAXReader saxReader = new SAXReader();  Document document=null;  try{  URL xmlpath = this.getClass().getClassLoader().getResource(filename);  document = saxReader.read(xmlpath);  Map<String,String> nsMap = new HashMap<String,String>();  nsMap.put("ns","http://www.springframework.org/schema/beans");//加入命名空间  XPath xsub = document.createXPath("//ns:beans/ns:bean");//创建beans/bean查询路径  xsub.setNamespaceURIs(nsMap);//设置命名空间  List<Element> beans = xsub.selectNodes(document);//获取文档下所有bean节点  for(Element element: beans){  String id = element.attributeValue("id");//获取id属性值  String clazz = element.attributeValue("class"); //获取class属性值  BeanDefinition beanDefine = new BeanDefinition(id, clazz);  beanDefines.add(beanDefine);  }  }catch(Exception e){  e.printStackTrace();  }  }  } |

# 第四讲 Spring的三种实例化Bean的方式

## 4.1 三种实例化bean的方式

### 4.1.1 使用类构造器实例化

|  |
| --- |
| <bean id=“orderService" class="cn.itcast.OrderServiceBean"/> |

### 4.1.2 使用静态工厂方法实例化

|  |
| --- |
| <bean id="personService" class="cn.itcast.service.OrderFactory" factory-method="createOrder"/>  public class OrderFactory {  public **static** OrderServiceBean createOrder(){  return new OrderServiceBean();  }  } |

### 4.1.3 使用实例工厂方法实例化:

|  |
| --- |
| <bean id="personServiceFactory" class="cn.itcast.service.OrderFactory"/>  <bean id="personService" factory-bean="personServiceFactory" factory-method="createOrder"/>  public class OrderFactory {  public OrderServiceBean createOrder(){  return new OrderServiceBean();  }  } |

# 第五讲 配置Spring管理的bean的作用域

## 5.1 Bean的作用域

.singleton

在每个Spring IoC容器中一个bean定义只有一个对象实例。默认情况下会在容器启动时初始化bean，但我们可以指定Bean节点的lazy-init=“true”来延迟初始化bean，这时候，只有第一次获取bean会才初始化bean。如：

<bean id="xxx" class="cn.itcast.OrderServiceBean" lazy-init="true"/>

如果想对所有bean都应用延迟初始化，可以在根节点beans设置default-lazy-init=“true“，如下：

<beans default-lazy-init="true“ ...>

.prototype

每次从容器获取bean都是新的对象。

.request

.session

.global session

指定Bean的初始化方法和销毁方法

<bean id="xxx" class="cn.itcast.OrderServiceBean" init-method="init" destroy-method="close"/>