**IOC的别名：依赖注入(DI)**  
2004年，Martin Fowler探讨了同一个问题，既然IOC是控制反转，那么到底是“哪些方面的控制被反转了呢？”，经过详细地分析和论证后，他得出了答案：“获得依赖对象的过程被反转了”。控制被反转之后，获得依赖对象的过程由自身管理变为了由IOC容器主动注入。于是，他给“控制反转”取了一个更合适的名字叫做“依赖注入（Dependency Injection）”。他的这个答案，实际上给出了实现IOC的方法：注入。所谓依赖注入，就是由IOC容器在运行期间，动态地将某种依赖关系注入到对象之中。

所以，依赖注入(DI)和控制反转(IOC)是从不同的角度的描述的同一件事情，就是指**通过引入IOC容器，利用依赖关系注入的方式，实现对象之间的解耦**。

在Spring中，依赖注入（DI）模式实现了控制反转（IoC）原理。让我们通过一个例子来帮助理解依赖注入。我们先看到java版的例子，然后在此基础上加上spring的功能。就例子而言，是相当地简单。QuizMater接口暴露了popQuestion()方法。为了保持简单性，QuizMaster将只生成一个问题。

/\*\*

\* QuizMaster.java

\*/

package com.vaannila;

public interface QuizMaster {

public String popQuestion();

}

StrutsQuizMaster和SpringQuizMaster类实现了QuizMaster接口，它们各自生成struts和spring相关的问题。

/\*\*

\* StrutsQuizMaster.java

\*/

package com.vaannila;

public class StrutsQuizMaster implements QuizMaster {

@Override

public String popQuestion() {

return "Are you new to Struts?";

}

}

/\*\*

\* SpringQuizMaster.java

\*/

package com.vaannila;

public class SpringQuizMaster implements QuizMaster {

@Override

public String popQuestion() {

return "Are you new to Spring?";

}

}

有个QuizMasterService类向用户显示问题。QuizMasterService类关联了QuizMaster接口。

/\*\*

\* QuizMasterService.java

\*/

package com.vaannila;

public class QuizMasterService {

private QuizMaster quizMaster = new SpringQuizMaster();

public void askQuestion(){

System.out.println(quizMaster.popQuestion());

}

}

最后创建QuizProgram类来管理测验。

/\*\*

\* QuizProgram.java

\*/

package com.vaannila;

public class QuizProgram {

public static void main(String[] args) {

QuizMasterService quizMasterService = new QuizMasterService();

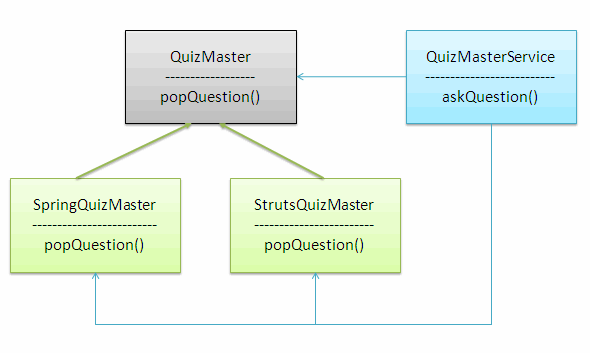
quizMasterService.askQuestion();

}

}

看上去相当的简单，我们创建了QuizMasterService类的实例，并且调用了askQuestion()方法。当你执行program，期望的“Are you new to Spring?”将从控制台打印出来。

再看看这个例子的类图。绿色箭头表示泛化，蓝色箭头表示关联。



 看的出来这种结构是紧密耦合在一起的。在QuizMasterService类中创建了QuizMaster的实例，如下所示。

private QuizMaster quizMaster = new SpringQuizMaster();

为了测验精通struts的人，我们需要修改QuizMasterService成这样：

private QuizMaster quizMaster = new StrutsQuizMaster();

因此耦合性是非常高的，这就是为什么要使用依赖注入来避免这种耦合。Spring框架提供了非常强大的容器来管理组件。容器就是基于控制反转（IoC）理念并且实现了依赖注入。这些组件仅仅需要选择一种接受资源的方式，容器会自动为组件推送资源。

下面我们替换为QuizMasterService类直接创建QuizMaster对象的方式，让容器来承担这项工作。替换了硬编码，允许容器来注入需要的依赖项。

注入依赖使用setter或构造方法注入。下面看看怎样使用setter来注入。

/\*\*

\* QuizMasterService.java

\*/

package com.vaannila;

public class QuizMasterService {

private QuizMaster quizMaster;

public void setQuizMaster(QuizMaster quizMaster) {

this.quizMaster = quizMaster;

}

public void askQuestion(){

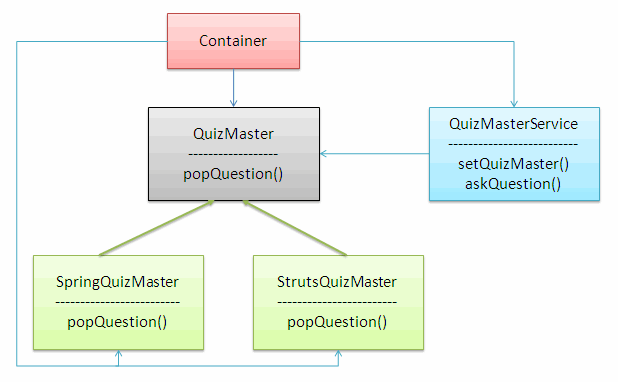
System.out.println(quizMaster.popQuestion());

}

}

QuizMaster的值使用setQuizMaster()方法设置。在QuizMasterService类中，QuizMaster对象没有实例化，但仍然来访问它。通常这样做会抛出NullPointerException异常，但是容器已经实例化了这个对象，因此能很好的工作。

再作了这些改变之后，例子的类图就变成这样了。



 图中多了容器，它帮助注入依赖项。

在beans.xml文件中的beans配置：

<?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.xsd">

<bean id="springQuizMaster" class="com.vaannila.SpringQuizMaster"></bean>

<bean id="strutsQuizMaster" class="com.vaannila.StrutsQuizMaster"></bean>

<bean id="quizMasterService" class="com.vaannila.QuizMasterService">

<property name="quizMaster">

<ref local="springQuizMaster" />

</property>

</bean>

</beans>

定义每个bean使用bean标签。bean标签的id属性是获取这个bean的逻辑名，class属性表示了真实的bean类。property标签指向bean的属性。使用setter注入一个bean需要使用ref标签。

SpringQuizMaster的引用被注入到QuizMasterbean。当我们执行这个例子，控制台打印出“Are you new to Spring?”。

为了使QuizMaster问一个struts相关的问题，仅仅需要改变ref标签的指向。

<bean id="quizMasterService" class="com.vaannila.QuizMasterService">

<property name="quizMaster">

<ref local="strutsQuizMaster" />

</property>

</bean>

依赖注入降低了组件之间的耦合。

要执行这个例子需要把如下的jar文件加大classpath。

antlr-runtime-3.0  
commons-logging-1.0.4  
org.springframework.asm-3.0.0.M3  
org.springframework.beans-3.0.0.M3  
org.springframework.context-3.0.0.M3  
org.springframework.context.support-3.0.0.M3  
org.springframework.core-3.0.0.M3  
org.springframework.expression-3.0.0.M3