Spring – day01

学习目标

（1）能够知道解耦的演变过程

（2）能够清晰描述出IOC相关思想

（3）能够搭建spring的开发环境

（4）能够使用xml的方式完成依赖注入配置

学习指南

spring的概述

程序的耦合及解耦

IOC概念和spring中的IOC（重点）

依赖注入（Dependency Injection）（重点）

# 第一堂课

**本节知识点**

spring概述

spring发展历程

spring的优势

spring的体系结构

编写jdbc的工程代码用于分析程序的耦合

编译期依赖

**本节目标**

能够说出spring框架的优点

描述出spring框架的体系结构

知道代码编译期的依赖

视频时间：

30分52秒

## spring课程四天安排

### 知识概述

介绍spring课程安排，4天内容大概描述

### 视频详情



### 总结与补充

spring共四天

第一天：spring框架的概述以及spring中基于XML的IOC配置

第二天：spring中基于注解的IOC和ioc的案例

第三天：spring中的aop和基于XML以及注解的AOP配置

第四天：spring中的JdbcTemlate以及Spring事务控制

### 课堂提问与练习

无

### 习题答案

无

## 今日课程内容介绍

### 知识概述

当日内容的课程内容概述

### 视频详情



### 总结与补充

1、spring的概述

spring是什么

spring的两大核心

spring的发展历程和优势

spring体系结构

2、程序的耦合及解耦

曾经案例中问题

工厂模式解耦

3、IOC概念和spring中的IOC

spring中基于XML的IOC环境搭建

4、依赖注入（Dependency Injection）

5、作业：

### 课堂提问与练习

无

### 习题答案

无

## spring概述

### 知识概述

Spring框架的概述

### 视频详情



### 总结与补充

**spring是什么？**

Spring是分层的 Java SE/EE应用 full-stack 轻量级开源框架，以 IoC（Inverse Of Control：

反转控制）和 AOP（Aspect Oriented Programming：面向切面编程）为内核，提供了展现层 SpringMVC 和持久层 Spring JDBC 以及业务层事务管理等众多的企业级应用技术，还能整合开源世界众多著名的第三方框架和类库，逐渐成为使用最多的Java EE 企业应用开源框架。

### 课堂提问与练习

无

### 习题答案

无

## spring发展历程

### 知识概述

Spring框架的发展历史，及历史版本

### 视频详情



### 总结与补充

Spring 的发展历程

1997 年 IBM提出了EJB 的思想

1998 年，SUN制定开发标准规范 EJB1.0

1999 年，EJB1.1 发布

2001 年，EJB2.0 发布

2003 年，EJB2.1 发布

2006 年，EJB3.0 发布

Rod Johnson （ spring 之父）

Expert One-to-One J2EE Design and Development(2002)

阐述了 J2EE 使用EJB 开发设计的优点及解决方案

Expert One-to-One J2EE Development without EJB(2004)

阐述了 J2EE 开发不使用 EJB的解决方式（Spring 雏形）

7 2017 年 9 9 月份发布了 g spring 的最新版本 0 spring 5.0 通用版（GA）

### 课堂提问与练习

无

### 习题答案

无

## spring的优势

### 知识概述

Spring框架代理的优势有哪些

### 视频详情



### 总结与补充

spring 的优势

**方便解耦，简化开发**

通过 Spring提供的 IoC容器，可以将对象间的依赖关系交由 Spring进行控制，避免硬编码所造成的过度程序耦合。用户也不必再为单例模式类、属性文件解析等这些很底层的需求编写代码，可以更专注于上层的应用。

**AOP 编程的支持**

通过 Spring的 AOP 功能，方便进行面向切面的编程，许多不容易用传统OOP 实现的功能可以通过 AOP 轻松应付。

**声明式事务的支持**

可以将我们从单调烦闷的事务管理代码中解脱出来，通过声明式方式灵活的进行事务的管理，提高开发效率和质量。

**方便程序的测试**

可以用非容器依赖的编程方式进行几乎所有的测试工作，测试不再是昂贵的操作，而是随手可做的事情。

**方便集成各种优秀框架**

Spring可以降低各种框架的使用难度，提供了对各种优秀框架（Struts、Hibernate、Hessian、Quartz等）的直接支持。

**降低 JavaEE API 的使用难度**

Spring对 JavaEE API（如 JDBC、JavaMail、远程调用等）进行了薄薄的封装层，使这些 API 的使用难度大为降低。

**Java 源码是经典学习范例**

Spring的源代码设计精妙、结构清晰、匠心独用，处处体现着大师对Java 设计模式灵活运用以及对 Java技术的高深造诣。它的源代码无意是 Java 技术的最佳实践的范例。

### 课堂提问与练习

无

### 习题答案

无

## spring的体系结构

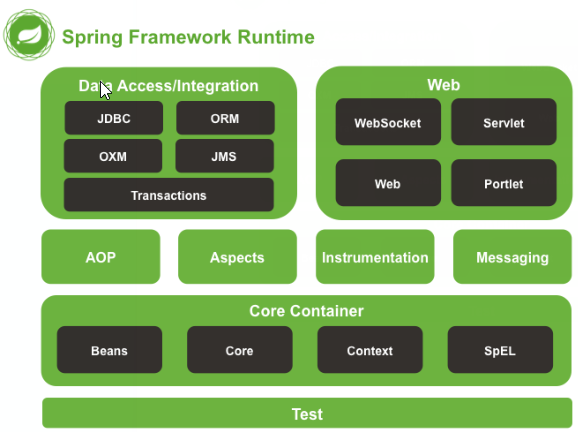
### 知识概述

看图讲解spring的体系结构

### 视频详情



### 总结与补充



### 课堂提问与练习

无

### 习题答案

无

## 编写jdbc的工程代码用于分析程序的耦合

### 知识概述

编写jdbc代码连接，分析代码中的耦合

### 视频详情



### 总结与补充



### 课堂提问与练习

无

### 习题答案

无

## 编译期依赖

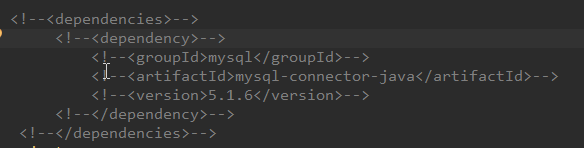
### 知识概述

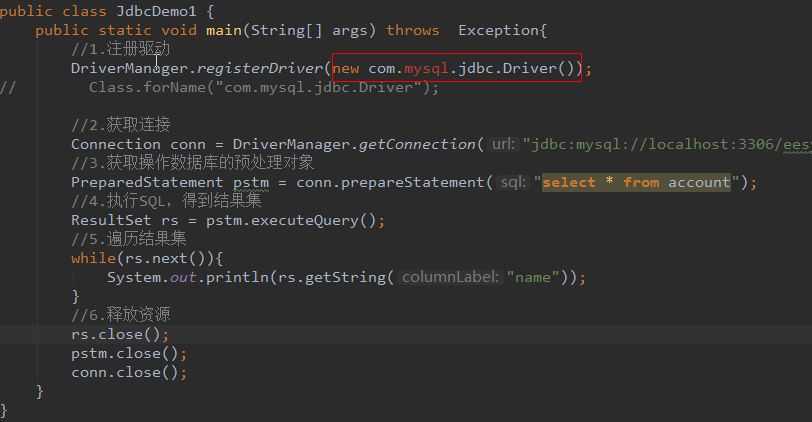
代码耦合中的编译器依赖分析

### 视频详情



### 总结与补充





找不到依赖的情况下，是编译不过去的。存在耦合。

### 课堂提问与练习

无

### 习题答案

无

# 第二堂课

**本节知识点**

2.1程序的耦合和解耦的思路分析1

2.2曾经代码中的问题分析

2.3编写工厂类和配置文件

2.4工厂模式解耦

**本节目标**

能够清楚程序中的耦合及解决耦合的思路

能够知道当前代码存在的问题

能够知道为什么要创建工厂类来创建对象并编写工厂类中的代码

视频时间：

35分55秒

## 程序的耦合和解耦的思路分析1

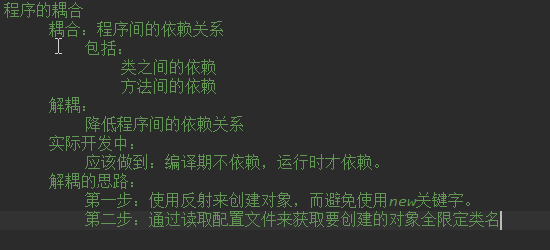
### 知识概述

讲解解耦合的思路

### 视频详情



### 总结与补充



### 课堂提问与练习

* 解决耦合的思路？

### 习题答案

第一步：使用反射来创建对象，而避免使用new关键字。

第二步：通过读取配置文件来获取要创建的对象全限定类名

## 曾经代码中的问题分析

### 知识概述

分析直接依赖new的问题

### 视频详情



### 总结与补充

创建业务层接口和实现类

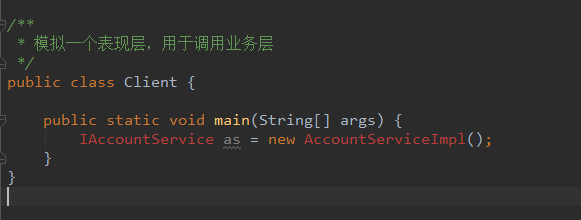




创建持久层接口和实现类



模拟表现层



### 课堂提问与练习

无

### 习题答案

无

## 编写工厂类和配置文件

### 知识概述

解决耦合第一步，编写工厂类及配置文件

### 视频详情

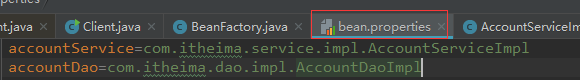


### 总结与补充

创建BeanFactory类

|  |
| --- |
| 一个创建Bean对象的工厂  Bean：在计算机英语中，有可重用组件的含义。  JavaBean：用java语言编写的可重用组件。  javabean > 实体类  它就是创建我们的service和dao对象的。  第一个：需要一个配置文件来配置我们的service和dao  配置的内容：唯一标识=全限定类名（key=value)  第二个：通过读取配置文件中配置的内容，反射创建对象  我的配置文件可以是xml也可以是properties |

创建一个bean.properties



### 课堂提问与练习

### 习题答案

## 工厂模式解耦

### 知识概述

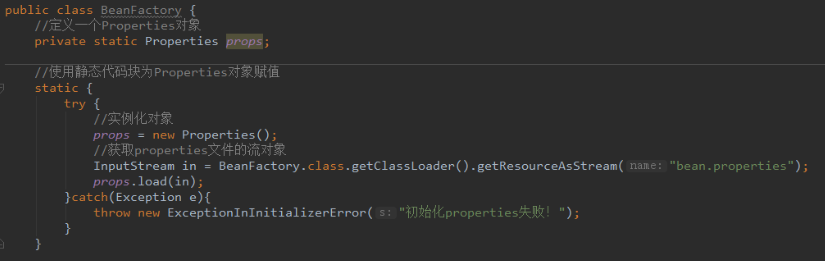
创建工厂来解决代码中的耦合

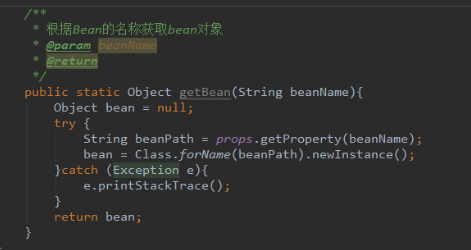
### 视频详情



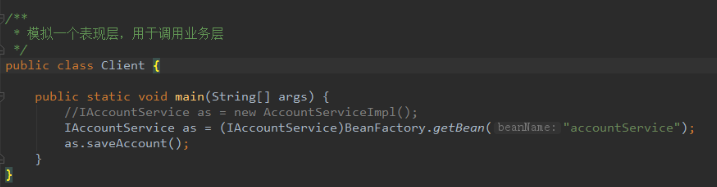
### 总结与补充

BeanFactory.java文件





测试类：



### 课堂提问与练习

无

### 习题答案

无

# 第三堂课

**本节知识点**

3.1分析工厂模式中的问题并改造

3.2工厂模式解耦的升级版

3.3 ioc的概念和作用

3.4 spring中的Ioc前期准备

**本节目标**

能够清楚编写工厂类的作用

能够清楚IOC解决的问题

知道spring的版本，jdk与tomcat的版本

视频时间：

25分55秒

## 分析工厂模式中的问题并改造

### 知识概述

目前工厂存在的问题，进行改造

### 视频详情



### 总结与补充

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

特别注意：最后有代码错误并没有解决，直接结束了视频，在视频（14工厂模式解耦的升级版.avi）中解决了代码错误，没有单独进行说明，关于总结，请看下个视频总结

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### 课堂提问与练习

无

### 习题答案

无

## 工厂模式解耦的升级版

### 知识概述

最终的工厂模式解耦的思路及代码实现

### 视频详情



### 总结与补充

BeanFactory代码

|  |
| --- |
| public class BeanFactory {  //定义一个Properties对象  private static Properties props;  //定义一个Map,用于存放我们要创建的对象。我们把它称之为容器  private static Map<String,Object> beans;  //使用静态代码块为Properties对象赋值  static {  try {  //实例化对象  props = new Properties();  //获取properties文件的流对象  InputStream in = BeanFactory.class.getClassLoader().getResourceAsStream("bean.properties");  props.load(in);  //实例化容器  beans = new HashMap<String,Object>();  //取出配置文件中所有的Key  Enumeration keys = props.keys();  //遍历枚举  while (keys.hasMoreElements()){  //取出每个Key  String key = keys.nextElement().toString();  //根据key获取value  String beanPath = props.getProperty(key);  //反射创建对象  Object value = Class.forName(beanPath).newInstance();  //把key和value存入容器中  beans.put(key,value);  }  }catch(Exception e){  throw new ExceptionInInitializerError("初始化properties失败！");  }  }  /\*\*  \* 根据bean的名称获取对象  \* @param beanName  \* @return  \*/  public static Object getBean(String beanName){  return beans.get(beanName);  }  /\*\*  \* 根据Bean的名称获取bean对象  \* @param beanName  \* @return  public static Object getBean(String beanName){  Object bean = null;  try {  String beanPath = props.getProperty(beanName);  // System.out.println(beanPath);  bean = Class.forName(beanPath).newInstance();//每次都会调用默认构造函数创建对象  }catch (Exception e){  e.printStackTrace();  }  return bean;  }\*/  } |

测试代码：

AccountServiceImpl代码：

|  |
| --- |
| /\*\*  \* 账户的业务层实现类  \*/  public class AccountServiceImpl implements IAccountService {  // private IAccountDao accountDao = new AccountDaoImpl();  private IAccountDao accountDao = (IAccountDao)BeanFactory.getBean("accountDao");  // private int i = 1;  public void saveAccount(){  int i = 1;  accountDao.saveAccount();  System.out.println(i);  i++;  }  } |

测试代码：

|  |
| --- |
| public class Client {  public static void main(String[] args) {  //IAccountService as = new AccountServiceImpl();  for(int i=0;i<5;i++) {  IAccountService as = (IAccountService) BeanFactory.getBean("accountService");  System.out.println(as);  as.saveAccount();  }  }  } |

### 课堂提问与练习

无

### 习题答案

无

## ioc的概念和作用

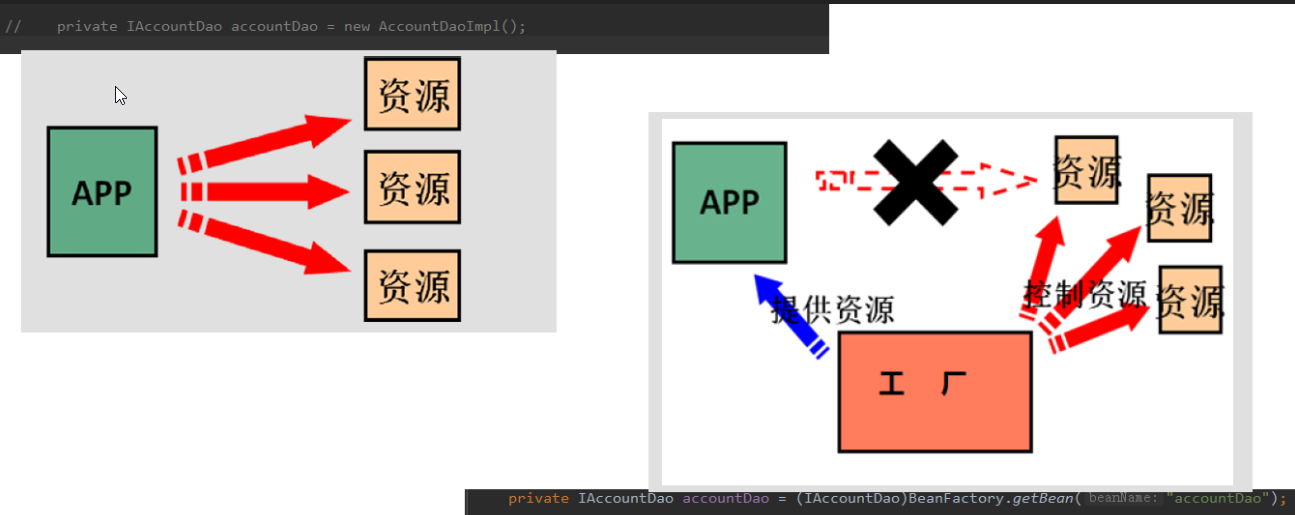
### 知识概述

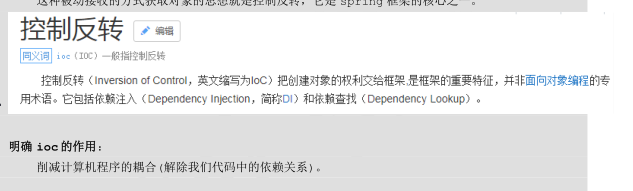
描述ioc的概念及解决的问题

### 视频详情



### 总结与补充





### 课堂提问与练习

Ioc的作用是什么？

### 习题答案

削减计算机程序的耦合(解除我们代码中的依赖关系)。

## spring基于XML的IOC环境搭建和入门

### 知识概述

搭建spring开发的环境，及环境要求

### 视频详情



### 总结与补充

案例的前期准备

本章我们使用的案例是，账户的业务层和持久层的依赖关系解决。在开始 spring 的配置之前，我们要先准备一下环境。由于我们是使用 spring 解决依赖关系，并不是真正的要做增删改查操作，所以此时我们没必要写实体类。并且我们在此处使用的是 java 工程，不是 java web 工程。

前期的准备：

官网：http://spring.io/

下载地址：

http://repo.springsource.org/libs-release-local/org/springframework/spring

解压:(Spring 目录结构:)

\* docs :API 和开发规范.

\* libs :jar 包和源码.

\* schema :约束.



我们上课使用的版本是 是 spring5.0.2 。

特别说明：

spring5版本是用 jdk8编写的，所以要求我们的 jdk 版本是 8及以上。

同时 tomcat的版本要求8.5及以上。

### 课堂提问与练习

无

### 习题答案

无

# 第四堂课

**本节知识点**

4.1 spring基于XML的IOC环境搭建和入门

4.2 ApplicationContext的三个实现类

4.3 BeanFactory和ApplicationContext的区别

**本节目标**

能够自主搭建IOC的环境，知道导入哪些jar包

清晰知道实现类的具体功能，区别

知道BeanFactory和ApplicationContext两个接口的区别

视频时间：

33分03秒

## spring基于XML的IOC环境搭建和入门

### 知识概述

编写转账案例代码

### 视频详情



### 总结与补充

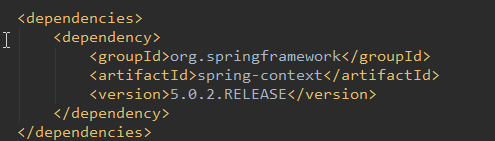
创建业务层接口和实现类

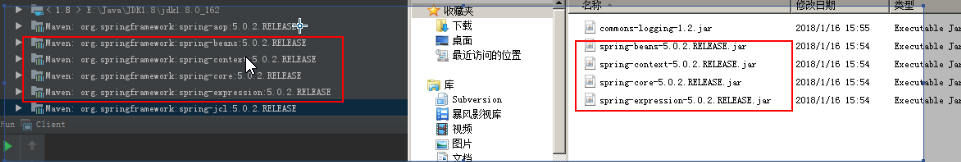


创建持久层接口和实现类

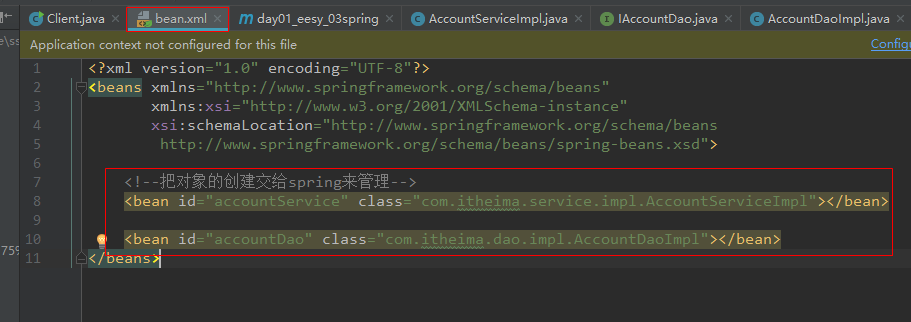


导入依赖：

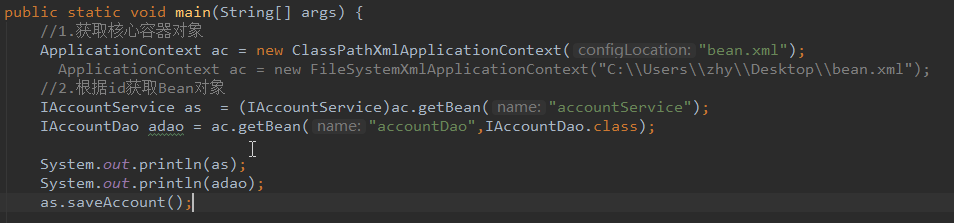




创建bean.xml文件，让spring来管理两个对象的创建



测试代码：



### 课堂提问与练习

无

### 习题答案

无

## ApplicationContext的三个实现类

### 知识概述

ClassPathXmlApplicationContext：

FileSystemXmlApplicationContext

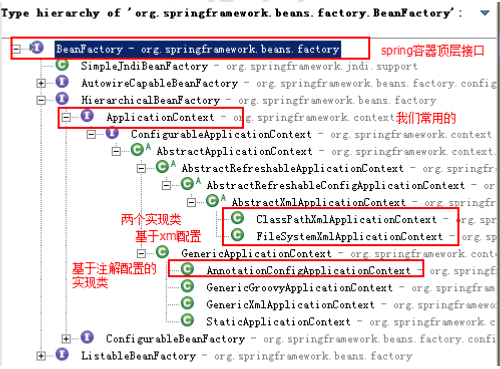
AnnotationConfigApplicationContext

三个实现类的讲解

### 视频详情



### 总结与补充



**ApplicationContext的三个常用实现类：**

ClassPathXmlApplicationContext：它可以加载类路径下的配置文件，要求配置文件必须在类路径下。不在的话，加载不了。(更常用)

FileSystemXmlApplicationContext：它可以加载磁盘任意路径下的配置文件(必须有访问权限）

AnnotationConfigApplicationContext：它是用于读取注解创建容器的，是明天的内容。

### 课堂提问与练习

无

### 习题答案

无

## BeanFactory和ApplicationContext的区别

### 知识概述

BeanFactory和ApplicationContext的区别，代码测试区别

### 视频详情



### 总结与补充

BeanFactory 和 和 ApplicationContext 的区别

BeanFactory 才是 Spring 容器中的顶层接口。

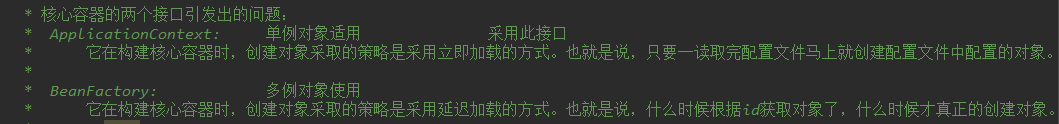
ApplicationContext 是它的子接口。

BeanFactory 和 ApplicationContext 的区别：

创建对象的时间点不一样。

ApplicationContext：只要一读取配置文件，默认情况下就会创建对象。

BeanFactory：什么使用什么时候创建对象。



### 课堂提问与练习

无

### 习题答案

无

# 第五堂课

**本节知识点**

5.1 spring中bean的细节之三种创建Bean对象的方式

5.2 spring中bean的细节之作用范围

5.3 spring中bean的细节之生命周期

**本节目标**

清楚spring实例化对象的三种方式

能够描述bean的五种作用范围

能够知道bean的单例和多例的生命周期

视频时间：

42分27秒

## spring中bean的细节之三种创建Bean对象的方式

### 知识概述

Bean的创建方式

无参构造，静态工厂，实例工厂

### 视频详情



### 总结与补充

第一种方式：使用默认无参构造函数

<!--在默认情况下：它会根据默认无参构造函数来创建类对象。如果 bean 中没有默认无参构造函数，将会创建失败。-->

<bean id="accountService" class="com.itheima.service.impl.AccountServiceImpl"/>





### 课堂提问与练习

无

### 习题答案

无

## spring中bean的细节之作用范围

### 知识概述

讲解spring中的bean的作用范围，scope的值选择

### 视频详情



### 总结与补充

作用：

用于配置对象让 spring 来创建的。

默认情况下它调用的是类中的无参构造函数。如果没有无参构造函数则不能创建成功。

属性：

id：给对象在容器中提供一个唯一标识。用于获取对象。

class：指定类的全限定类名。用于反射创建对象。默认情况下调用无参构造函数。

scope：指定对象的作用范围。

\* singleton :默认值，单例的.

\* prototype :多例的.

\* request :WEB 项目中,Spring 创建一个 Bean 的对象,将对象存入到 request 域中.

\* session :WEB 项目中,Spring 创建一个 Bean 的对象,将对象存入到 session 域中.

\* global session :WEB 项目中,应用在 Portlet 环境.如果没有 Portlet 环境那么

globalSession 相当于 session.

init-method：指定类中的初始化方法名称。

destroy-method：指定类中销毁方法名称。

### 课堂提问与练习

无

### 习题答案

无

## spring中bean的细节之生命周期

### 知识概述

详细讲解bean单例和多例的生命周期

### 视频详情



### 总结与补充

bean对象的生命周期

单例对象

出生：当容器创建时对象出生

活着：只要容器还在，对象一直活着

死亡：容器销毁，对象消亡

总结：单例对象的生命周期和容器相同

多例对象

出生：当我们使用对象时spring框架为我们创建

活着：对象只要是在使用过程中就一直活着。

死亡：当对象长时间不用，且没有别的对象引用时，由Java的垃圾回收器回收

### 课堂提问与练习

无

### 习题答案

无

# 第六堂课

**本节知识点**

6.1 spring的依赖注入

6.2 构造函数注入

6.3 构造函数注入

6.4 set方法注入

6.5 注入集合数据

6.6 课程总结

**本节目标**

能够描述什么是依赖注入

能够使用编写构造函数注入的代码

能够编写set方法注入的代码

能够掌握集合类型注入的方式

视频时间：

45分03秒

## spring的依赖注入

### 知识概述

讲解依赖注入的概念，及spring中注入的方式讲解

### 视频详情



### 总结与补充

依赖注入的概念

依赖注入：Dependency Injection。它是 spring 框架核心 ioc 的具体实现。

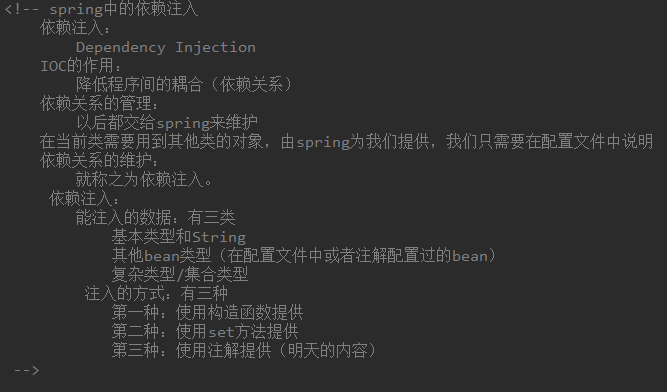
我们的程序在编写时，通过控制反转，把对象的创建交给了 spring，但是代码中不可能出现没有依赖的情况。

ioc 解耦只是降低他们的依赖关系，但不会消除。例如：我们的业务层仍会调用持久层的方法。

那这种业务层和持久层的依赖关系，在使用 spring 之后，就让 spring 来维护了。

简单的说，就是坐等框架把持久层对象传入业务层，而不用我们自己去获取。

以下是代码中的注释：



### 课堂提问与练习

什么是依赖注入？

### 习题答案

以后都交给spring来维护，在当前类需要用到其他类的对象，由spring为我们提供，我们只需要在配置文件中说明，依赖关系的维护：就称之为依赖注入。

## 构造函数注入

### 知识概述

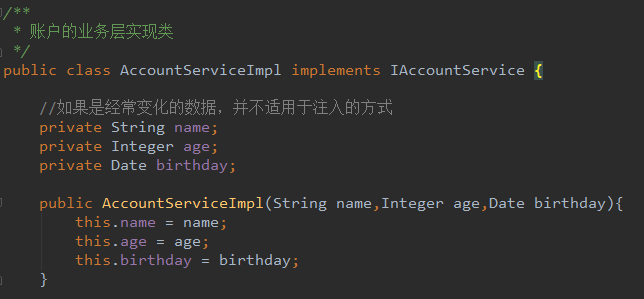
使用构造器注入，日期类型注入方式引出

### 视频详情

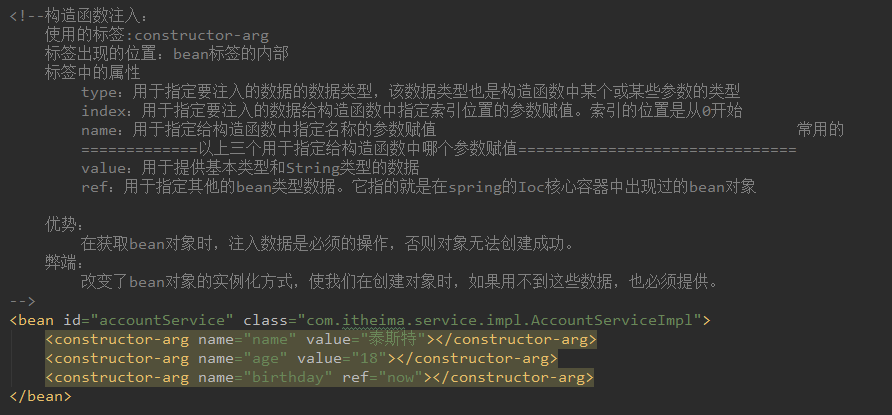


### 总结与补充

在AccountServiceImpl提供参数和构造函数



bean.xml文件中的配置



### 课堂提问与练习

无

### 习题答案

无

## 构造函数注入(2)

### 知识概述

解决日期类型的注入

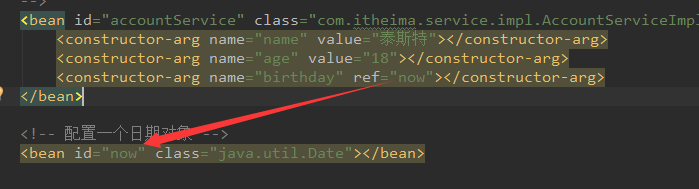
### 视频详情



### 总结与补充

解决日期类型

bean.xml文件中的配置



### 课堂提问与练习

无

### 习题答案

无

## set方法注入

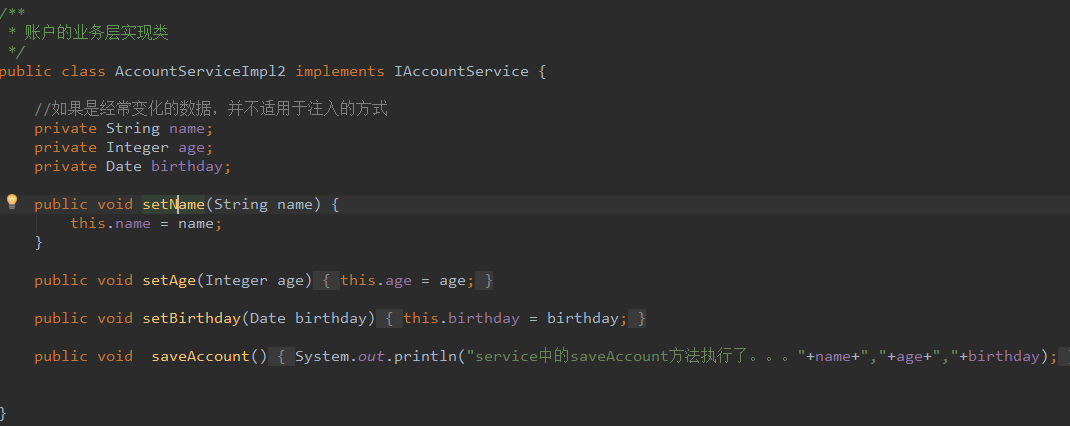
### 知识概述

### 视频详情

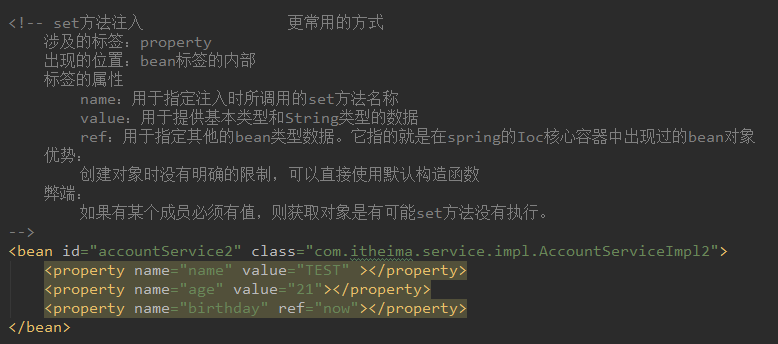


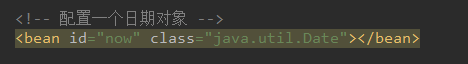
### 总结与补充

AccountServiceImpl2



Bean.xml文件





### 课堂提问与练习

无

### 习题答案

无

## 注入集合数据

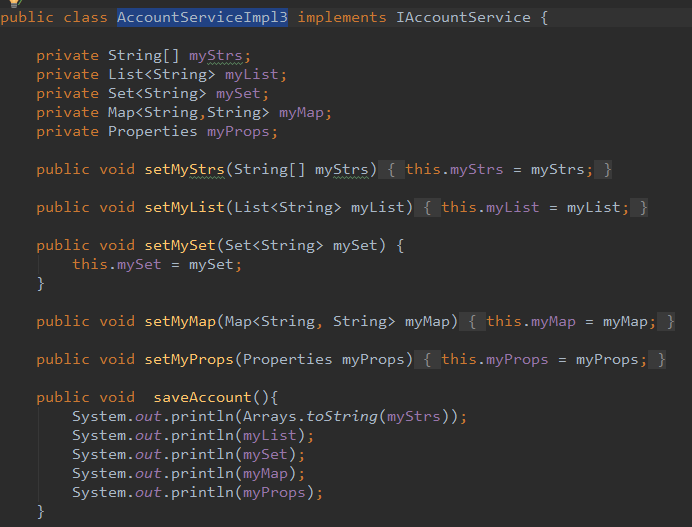
### 知识概述

### 视频详情



### 总结与补充

编写AccountServiceImpl3，提供集合相关的类型



Bean.xml





### 课堂提问与练习

无

### 习题答案

无

## 课程总结

### 知识概述

### 视频详情



### 总结与补充

讲解当天的重点，告知预习的内容，提供笔记进行预习

### 课堂提问与练习

无

### 习题答案

无