Spring框架

什么是Spring

Spring是分层的 JavaSE及JavaEE应用于全栈 的轻量级开源框架以IOc(Inverse of Contorl;

控制反转/反转控制)和AOP（Aspact Oriented Programming;面向切面编程）为核心提供了表现层MVC和持久层Spring JDBC以及业务层事务管理等众多模块的企业级应用技术还能整合到开源世界中众多著名的第三方框架和类库逐渐成为使用最多的JAVAEEq企业应用开源框架

Struts2 Spring Hibernate

Springmvc Spring Mybatis

Spring的本质是管理软件的对象即创建对象和维护对象之间的关系

Spring的发展历程

1997年 IBM提出了EJB的思想，开发的时候成本太高了，如果有新的技术维护起来会很麻烦

1998年 Sun指定了开发标准规范EJB1.0

1999年EJB1.1发布

2001年EJB2.0发布

2003年EJB2.1发布

2005年EJB3.0发布

Struts 现在逐渐被Springmvc

Struts的源码中有重大漏洞而且学习成本高于Springmv

Hibernate 逐渐被替换成Mybatis

Hibernate本身是生成SQL语句

开发的时候是方便但是开发的时候由于是自动生成的所以代码是死板的

无法优化sql代码

Spirng的本质是管理软件中的对象 即创建对象和维护对象之间的关系

Spring的优势

耦合是指我们早期会用new来创建对象，new会造成程序的依赖关系提高，造成耦合性更高，我们通过Spring的IoC来让Spring控制，便面编码所造成的过度程序耦合用户也不必要再为较为底层的需求编写代码可以更专注于上次的应用

AOP变成的支持

通过Spring的AOP功能 翻遍进行面向切面的变成 许多不容易用

传统OOP实现的功能可以用AOP请求的应付

声明式事务的支持

可以将我们从单调范孟的事务管理大妈中解脱出来通过声明的方式来更加灵活的进行

事务的管理提高开发效率和质量

方便程序的测试

可以用非容器所依赖的变成方式进行所有的测试工作测试不再是昂贵的操作二十随手可做的事情

方便继承各种优秀的框架

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

降低了JavaEE的使用难度

Spring对JavaEE api如（JDBC ,JAVAMail ,远程调用等）进行了薄薄的封装曾使这些API的使用难度大卫的降低

Spring框架是经典的学习范例

Spring的源码设计精妙结构企管系匠心独用 处处体现了大师对Java设计模式灵活运用

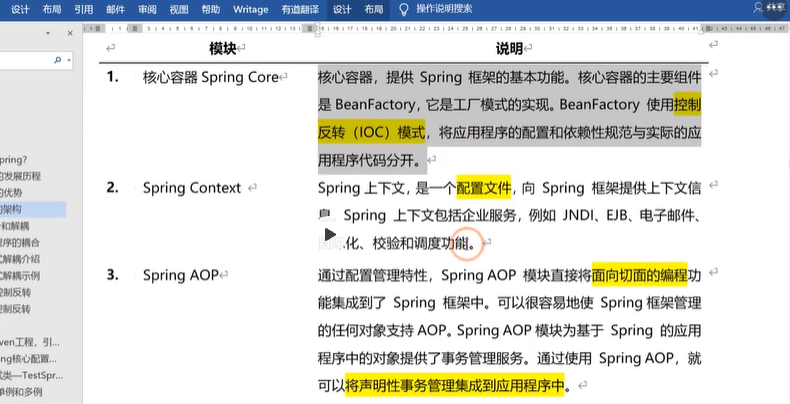
以及对Java技术的高深造纸它的源码无意是Java技术最佳实践的范例

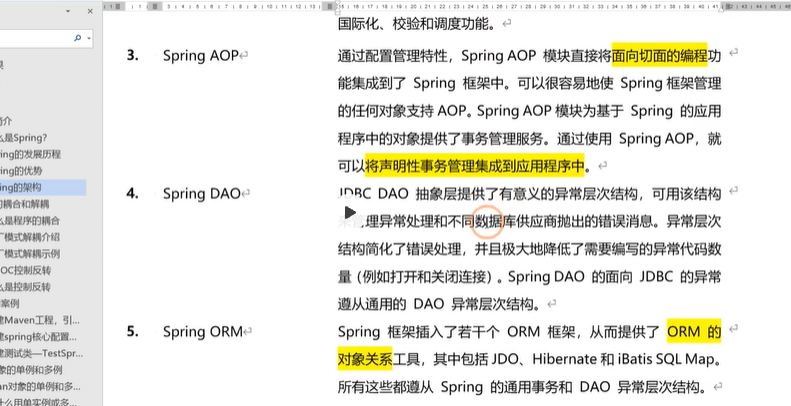
Spring的架构



整合一切优秀的框架然后对外提供一个统一的服务 Spring模块构建在核心容器之上核心容器定义了创建 配置 和管理bean的方式 如下图所示

组成Spring框架的每个模块或者组件都可以单独存在或者与其他一个或多个模块的联合实现每个模块表现如下







程序中的耦合和解耦

什么是程序的耦合

耦合性也叫做耦合度 是对模块之间关联程度的度量耦合的强弱取决于模块间

接口的复杂性调用模块的方式以及通过界面传送数据的多少模块间的耦合度是指

模块之间的依赖关系包括控制关系调用关系数据传递关系 模块联系越多其耦合性越强

同时表明其独立性越差（降低耦合性可以提高其独立性）耦合性存在各个领域

而非软件设计中独有的 但是我们只是讨论软件工程中的耦合

总结：在软件工程中耦合值得就是对象之间的依赖关系

对象之间的依赖关系越高耦合度就越高对象之间的耦合性越高维护的成本就越高

因此对象的设计应使用类和构建之间的耦合最小

降低程序之间的依赖程度即降低程序之间的耦合度的过程就叫做解耦

列入早期的JDBC操作中在注册数据库驱动的时候为什么采用的是Class.forName

的方式而不是采用DriverManager.registerDriver的方式

Public class TestJDBC{

Public static mian(String[] args) throws Exception{



}

}

DriverManager.registerDriver(new Driver());

会注册两次

工厂模式解耦的介绍

在实际的开发中可以将三层（表现层，业务层，持久层）的对象

都使用配置文件配置起来当启动服务器加载应用的时候可以通过工厂

读取配置文件根据配置文件中的配置将这些对象创建出来在接下来

使用的时候直接拿过来就可以了

那么这个负责读取配置文件更具配置文件创建并且返回这些 对象的类就是工厂

可以通过工厂+接口+配置文件的方式接触程序中的耦合

工厂模式解耦示例

解耦程序的编写步骤

创建一个Maven的java工程（java16-spring）

2.创建持久层的接口和接口的实现类

Bean 计算机语言中可以重用的组件

JavaBean是使用java程序编写的可宠用组件service/dao/pojo

业务bean 用于处理业务逻辑 service dao

实体bean 用于封装数据 列入：为了封装员工信息二提供emp类 就是实体bean

1. 需要提供配置xml properties

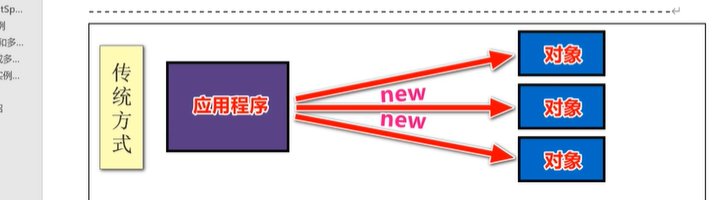
Spring IOC控制反转

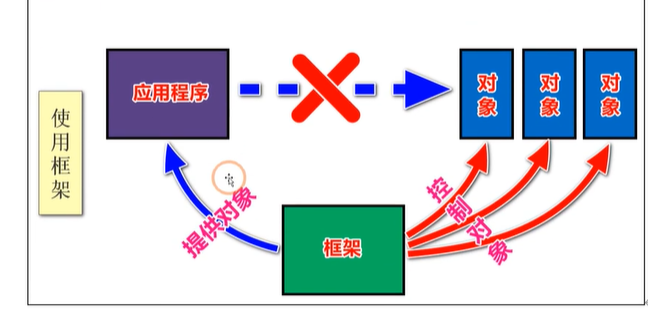
什么是控制反转

IOC Inverse of Control 控制反转即把创建对象的权力交给框架

也就是指将对象的创建对象的存储 ，对象的管理交给了Spring容器

Spring容器是Spring一个核心模块用于管理对象 底层可以理解为是一个map的集合





再次之间当需要对象的时候 通常是利用new关键按此创建一个对象

获取service接口的子列实例

这里使用new对象的方式造成了程序之间耦合性提升

Private EmpService service=(EmpService) BeamFactory .getBean(“EmpService“);

只需要将类提前配置在配置文件中就可以将对象的创建交给框架来做当需要对象时不需要自己

创建而是通过框架直接获取就行了省去了new对象的过程自然就是降低了类

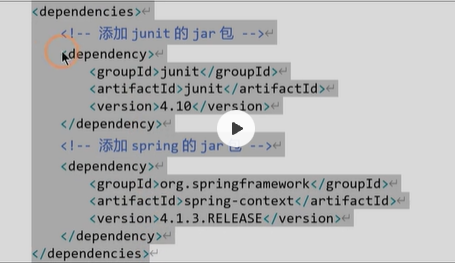
和类之间的依赖关系也就是耦合性

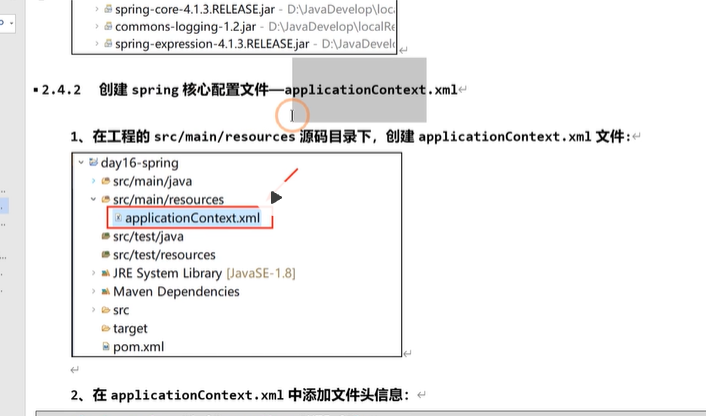
入门案例

下面是使用Spring的IOC解决程序之间的耦合

2.引入Junit 、Spring的jar包 子啊maven工程的pom.xml文件的根标签

Project内添加如下配置







Bean对象的单例和多例

在Spring容器中管理的Bean对象的作用域 可以通过Scope属性或用相关

注解指定其作用域，最常用是singleton（单例）或prototype多例其含义如下

Singleton单实例 是默认值这个作用域表示的对象具备全局的唯一性

当把一个bean定义设置scope为singleton作用域时那么Spring IOC

容器指挥创建该bean定义的唯一实列也就是说整个Spring IOC容器指挥创建当前类的一个对象

这个单一实例会被存储到单例缓存 并且正对该Bean的后续请求和引用都会

将返回被缓存的唯一的这个对象实例

Singleton负责对象的创建，初始化，销毁

Prototype多实例 这个作用表示的对象每次都会去创建新的对象

当把一个bean定义设置scope为singleton作用域的时候Spring IOC容器会在

每一次获取当前

Spring的DI依赖注入

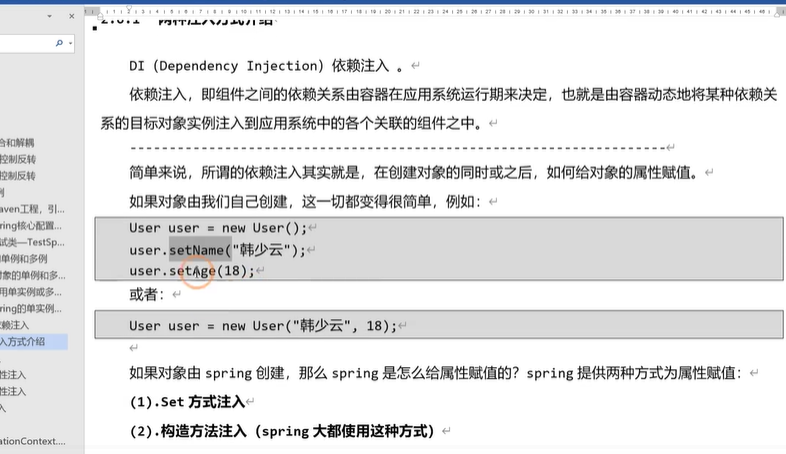
依赖注入 就是组件之间的依赖关系由容器在应用系统运行期来据欸的那个也就是

由容器动态地将某种依赖关系的目标对象实例主流到应用系统中的各个

关联的组件之中

简单的来说 所谓的依赖注入其实就是子啊创建对象的同时或之后 如何给对象的属性赋值

如果对象有我们自己创建这一切就会变得很贱例如



方式注入

普通属性注入

需求通过Spring创建user实例 并未user实例的name和agre属性赋值

