



《一头扎进 Spring4》视频教程》

第一章 问候 Spring4 他大爷

Java1234_小锋

扣扣:527085608

官网网站: <http://www.java1234.com>

Java1234 官方群 1, 2, 3, 4, 5, 6, 7, 8: (已满)

Java1234 官方群 9: 344186241

第一节：Spring 简介

Spring 作者：Rod Johnson;

官方网站：<http://spring.io/>

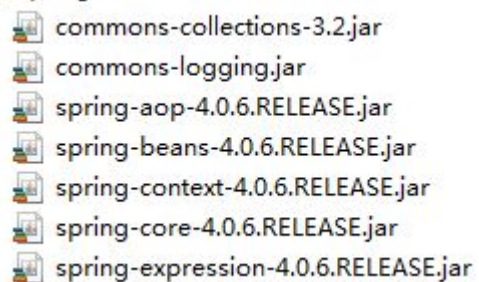
最新开发包及文档下载地址：<http://repo.springsource.org/libs-release-local/org/springframework/spring/>

核心思想：IOC 控制反转；AOP 面向切面；

介绍：百度百科；

第二节：Spring4 版 Hello World 实现

核心 jar 包：



- commons-collections-3.2.jar
- commons-logging.jar
- spring-aop-4.0.6.RELEASE.jar
- spring-beans-4.0.6.RELEASE.jar
- spring-context-4.0.6.RELEASE.jar
- spring-core-4.0.6.RELEASE.jar
- spring-expression-4.0.6.RELEASE.jar



《一头扎进 Spring4》视频教程》

第二章 Spring 之 IOC 详解

Java1234_小锋

扣扣:527085608

官网网站: <http://www.java1234.com>

Java1234 官方群 1, 2, 3, 4, 5, 6, 7, 8: (已满)

Java1234 官方群 9: 344186241

第一节：spring ioc 简介

IOC（控制反转：Inverse of Control ），又称作 依赖注入，是一种重要的面向对象编程的法则来削减计算机程序的耦合问题，也是轻量级的 Spring 框架的核心。

第二节：spring ioc 实例讲解

第三节：装配一个 bean

第四节：依赖注入

- 1，属性注入；
- 2，构造函数注入；(通过类型；通过索引；联合使用)
- 3，工厂方法注入；(非静态工厂，静态工厂)
- 4，泛型依赖注入；(Spring4 整合 Hibernate4 的时候顺带讲)

第五节：注入参数

- 1，基本类型值；
- 2，注入 bean；
- 3，内部 bean；
- 4，null 值；
- 5，级联属性；
- 6，集合类型属性；

第六节：Spring 自动装配

通过配置 default-autowire 属性，Spring IOC 容器可以自动为程序注入 bean；默认是 no，不启用自动装配；
default-autowire 的类型有 byName,byType,constructor；
byName：通过名称进行自动匹配；
byType：根据类型进行自动匹配；
constructor：和 byType 类似，只不过它是根据构造方法注入而言的，根据类型，自动注入；

建议：自动装配机制慎用，它屏蔽了装配细节，容易产生潜在的错误；

第七节：方法注入

Spring bean 作用域默认是 单例 singleton； 可以通过配置 prototype ， 实现多例；

方法注入 lookup-method

第八节：方法替换

第九节：bean 之间的关系

- 1， 继承；
- 2， 依赖；
- 3， 引用；

第十节：bean 作用范围

- 1， singleton Spring ioc 容器中仅有一个 Bean 实例， Bean 以单例的方式存在；
- 2， prototype 每次从容器中调用 Bean 时， 都返回一个新的实例；
- 3， request 每次 HTTP 请求都会创建一个新的 Bean；
- 4， session 同一个 HTTP Session 共享一个 Bean；
- 5， global session 同一个全局 Session 共享一个 Bean， 一般用于 Portlet 应用环境；
- 6， application 同一个 Application 共享一个 Bean；



《一头扎进 Spring4》视频教程》

第三章 Spring 之 AOP 详解

Java1234_小锋

扣扣:527085608

官网网站: <http://www.java1234.com>

Java1234 官方群 1, 2, 3, 4, 5, 6, 7, 8: (已满)

Java1234 官方群 9: 344186241

第一节：AOP 简介

AOP 简介：百度百科；
面向切面编程(也叫面向方面编程)：Aspect Oriented Programming(AOP)，是软件开发中的一个热点，也是 Spring 框架中的一个重要内容。利用 AOP 可以对业务逻辑的各个部分进行隔离，从而使得业务逻辑各部分之间的耦合度降低，提高程序的可重用性，同时提高了开发的效率。
主要的功能是：日志记录，性能统计，安全控制，事务处理，异常处理等等。

第二节：Spring AOP 实例

- 1，前置通知；
- 2，后置通知；
- 3，环绕通知；
- 4，返回通知；
- 5，异常通知；



《一头扎进 Spring4》视频教程》

第四章 Spring 对 DAO 的支持

Java1234_小锋

扣扣:527085608

官网网站: <http://www.java1234.com>

Java1234 官方群 1, 2, 3, 4, 5, 6, 7, 8: (已满)

Java1234 官方群 9: 344186241

第一节：Spring 对 JDBC 的支持

- 1，配置数据源 dbcp；
- 2，使用 JdbcTemplate；
- 3，JdbcDaoSupport 的使用；
- 4，NamedParameterJdbcTemplate 的使用；支持命名参数变量；
`org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate`

第二节：Spring 对 Hibernate 的支持

后面 Spring 整合 Hibernate 的时候讲；



《一头扎进 Spring4》视频教程》

第五章 Spring 对事务的支持

Java1234_小锋

扣扣:527085608

官网网站: <http://www.java1234.com>

Java1234 官方群 1, 2, 3, 4, 5, 6, 7, 8: (已满)

Java1234 官方群 9: 344186241

第一节：事务简介

满足一下四个条件：

- 第一：原子性；
- 第二：一致性；
- 第三：隔离性；
- 第四：持久性；

第二节：编程式事务管理

Spring 提供的事务模版类：`org.springframework.transaction.support.TransactionTemplate`
事务管理器：`org.springframework.jdbc.datasource.DataSourceTransactionManager`

第三节：声明式事务管理

- 1，使用 XML 配置声明式事务；
- 2，使用注解配置声明式事务；

第四节：事务传播行为

事务传播行为：Spring 中，当一个 service 方法调用另外一个 service 方法的时候，因为每个 service 方法都有事务，这时候就出现了事务的嵌套；由此，就产生了事务传播行为；
在 Spring 中，通过配置 Propagation，来定义事务传播行为；

- PROPAGATION_REQUIRED--支持当前事务，如果当前没有事务，就新建一个事务。这是最常见的选择。
- PROPAGATION_SUPPORTS--支持当前事务，如果当前没有事务，就以非事务方式执行。
- PROPAGATION_MANDATORY--支持当前事务，如果当前没有事务，就抛出异常。
- PROPAGATION_REQUIRES_NEW--新建事务，如果当前存在事务，把当前事务挂起。
- PROPAGATION_NOT_SUPPORTED--以非事务方式执行操作，如果当前存在事务，就把当前事务挂起。
- PROPAGATION_NEVER--以非事务方式执行，如果当前存在事务，则抛出异常。

```
<tx:attributes>
  <tx:method name="insert*" propagation="REQUIRED" />
  <tx:method name="update*" propagation="REQUIRED" />
  <tx:method name="edit*" propagation="REQUIRED" />
  <tx:method name="save*" propagation="REQUIRED" />
  <tx:method name="add*" propagation="REQUIRED" />
  <tx:method name="new*" propagation="REQUIRED" />
  <tx:method name="set*" propagation="REQUIRED" />
  <tx:method name="remove*" propagation="REQUIRED" />
  <tx:method name="delete*" propagation="REQUIRED" />
  <tx:method name="change*" propagation="REQUIRED" />
  <tx:method name="get*" propagation="REQUIRED" read-only="true" />
  <tx:method name="find*" propagation="REQUIRED" read-only="true" />
  <tx:method name="load*" propagation="REQUIRED" read-only="true" />
  <tx:method name="*" propagation="REQUIRED" read-only="true" />
</tx:attributes>
```