Neuedu

东软睿道内部公开

文件编号： D000-

Spring框架技术

版本：3.6.0-0.0.0

第1章 Spring入门

本章教学目标

- 了解Spring框架的基本概念；
- 理解Spring框架的体系结构；
- 掌握如何创建Spring项目；
- 掌握Spring装配机制
- 掌握使用Java和XML两种方式装配Bean
- 掌握使用JUnit测试运行Spring项目

本章内容

Neuedu

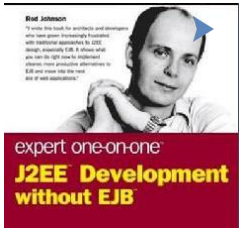
节	知识点	掌握程度	难易程度	教学形式	对应微课
Spring概述	Spring简介	了解	普通	线下	Spring简介
	Spring框架特征	了解	普通	线下	Spring框架特征
	Spring框架特点	了解	普通	线下	Spring框架特点
	Spring框架优势	了解	普通	线下	Spring框架优势
	Spring框架版本介绍	了解	普通	线下	Spring框架版本介绍
	Spring体系结构	理解	普通	线下	Spring体系结构
Spring快速入门	创建第一个Spring项目	掌握	普通	线下	创建第一个Spring项目
	Spring装配机制介绍	掌握	普通	线下	Spring装配机制介绍
	使用Java配置类装配Bean	掌握	普通	线下	使用Java配置类装配Bean
	测试运行Java配置	掌握	普通	线下	测试运行Java配置
	使用XML装配Bean	掌握	普通	线下	使用XML装配Bean
	测试运行Xml配置	掌握	普通	线下	测试运行Xml配置
整合Junit进行单元测试	Spring整合JUnit进行单元测试	掌握	普通	线下	Spring整合JUnit进行单元测试

CONTENTS 目录

Spring简介

Neuedu

- Spring 是开源的轻量级的企业级Java应用程序开发框架，Spring 为简化企业级应用开发而生，于2003由Rod Johnson（罗德·约翰逊)创建。
  - Rod Johnson在2002年编著的《Expert one on one J2EE design and development》一书中，对Java EE 系统框架臃肿、低效、脱离现实的种种现状提出了质疑，并积极寻求探索革新，以此书为指导思想，他编写了interface21框架，这是一个力图冲破Java EE传统开发的困境，从实际需求出发，着眼于轻便、灵巧，易于开发、测试和部署的轻量级开发框架。Spring框架即以interface21框架为基础，经过重新设计，并不断丰富其内涵，于2004年3月24日，发布了1.0正式版。



他又推出了一部堪称经典的力作《Expert one-on-one J2EE Development without EJB》，该书在Java世界掀起了轩然大波，不断改变着Java开发者程序设计和开发的思考方式。在该书中，对EJB的各种笨重臃肿的结构进行了逐一的分析和否定，并分别以简洁实用的方式替换之。至此一战功成，成为一个改变Java世界的大师级人物。

- Spring致力于J2EE应用的各层的解决方案，而不是仅仅专注于某一层的方案。可以说Spring是企业应用开发的“一站式”选择，并贯穿表现层、业务层及持久层。然而，Spring并不想取代那些已有的框架，而是与它们无缝地整合。
- 官方网站： https://spring.io

Spring框架特征

Neuedu

- 轻量——从大小与开销两方面而言Spring都是轻量的。完整的Spring框架可以在一个大小只有1MB多的JAR文件里发布。
- 控制反转——Spring通过一种称作控制反转（IoC）的技术促进了低耦合。
- 面向切面——Spring提供了面向切面编程的丰富支持，允许通过分离应用的业务逻辑与系统级服务（例如事务、日志管理）进行内聚性的开发。
- 容器——Spring包含并管理应用对象的配置和生命周期，可以配置每个bean如何被创建。
- 框架——Spring可以将简单的组件配置、组合成为复杂的应用。

Spring框架特点

Neuedu

- 1.方便解耦，简化开发
  - 通过Spring提供的IoC容器，将对象之间的依赖关系交由Spring进行控制，避免硬编码所造成的过度程序耦合。
- 2.AOP编程的支持
  - 通过Spring提供的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。
- 3.声明式事务的支持
  - 在Spring中，我们可以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地进行事务的管理，提高开发效率和质量。
- 4.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，例如：Spring对Junit4支持，可以通过注解方便的测试Spring程序。
- 5.方便集成各种优秀框架
  - Spring不排斥各种优秀的开源框架，相反，Spring可以降低各种框架的使用难度，Spring提供了对各种优秀框架（如Struts,MyBatista、Hibernate、Hessian、Quartz）等的直接支持。
- 6.降低Java EE API的使用难度
  - Spring对很多难用的Java EE API（如JDBC，JavaMail，远程调用等）提供了封装，通过Spring的简易封装，这些Java EE API的使用难度大为降低。
- 7.Java 源码是经典学习范例
  - Spring的源码设计精妙、结构清晰、匠心独运，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，Spring框架源码无疑是Java技术的最佳实践范例，学习和研究Spring源码将会使你迅速提高自己的Java技术水平和应用开发水平。

Spring框架优势

Neuedu

- 1.低侵入式设计，代码污染极低；
- 2.独立于各种应用服务器，基于Spring框架的应用，可以真正实现Write Once,Run Anywhere的；
- 3.Spring的DI机制降低了业务对象替换的复杂性，提高了组件之间的解耦；
- 4.Spring的AOP支持允许将一些通用任务如安全、事务、日志等进行集中式管理，从而提供了更好的复用；
- 5.Spring的ORM和DAO提供了与第三方持久层框架的良好整合，并简化了底层的数据库访问；
- 6.Spring并不强制应用完全依赖于Spring，开发者可自由选用Spring框架的部分或全部；

Spring框架版本介绍

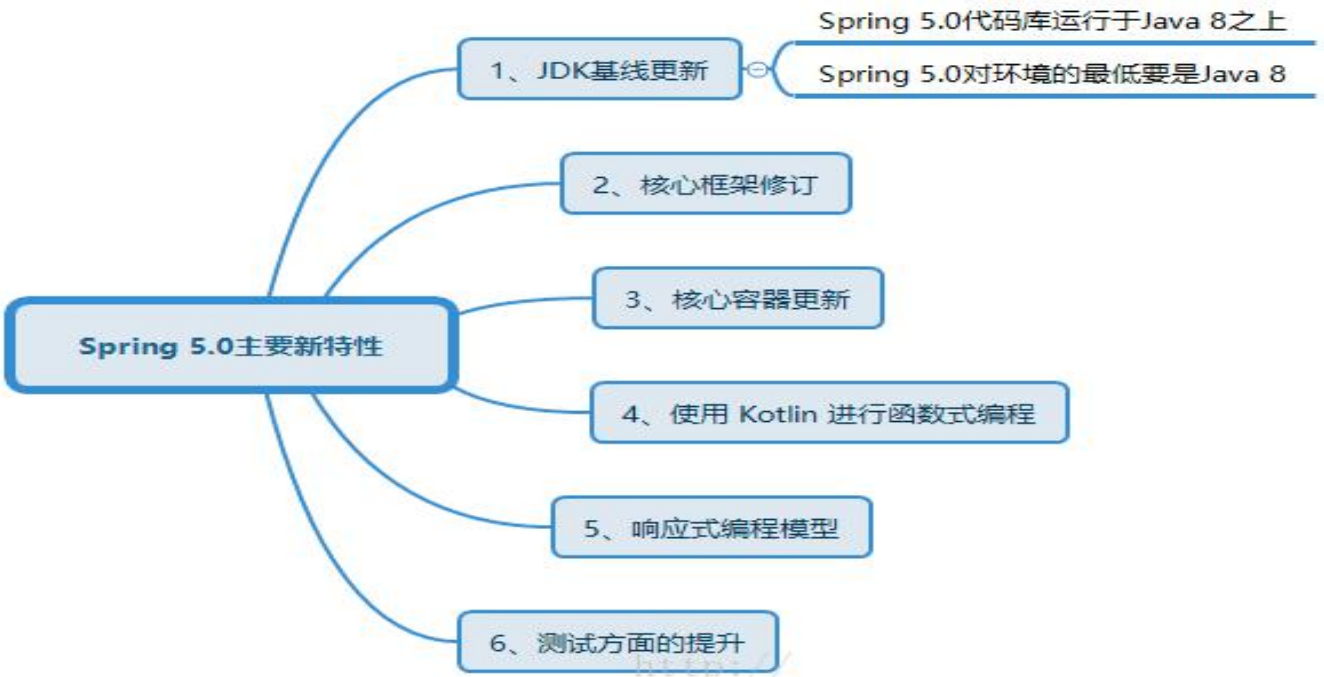
Neuedu

- Spring 1.x
  - 2004年3月24日Spring Framework 1.0 final正式发布，Spring 1.0当时只包含一个完整的项目，他把所有的功能都集中在一个项目中，其中包含了核心的Ioc、AOP，同时也包含了其他的诸多功能，例如：JDBC、Mail、ORM、事务、定时任务、Spring MVC等。
- Spring 2.x
  - Spring 2.0(2006/10)增加对注解的支持，支持了基于注解的配置。
- Spring 3.x
  - Spring 3.0(2009/12)支持了基于Java类的配置。
- Spring 4.x
  - 4.0.0（2013/11）、4.3.18(2016/6)
  - Spring 4.x全面支持Java 8.0，支持Lambda表达式的使用。
  - 核心容器增加：支持泛型的依赖注入、Map的依赖注入、Lazy延迟依赖的注入、List注入、Condition条件注解注入、对CGLib动态代理类进行了增强。
  - 支持了基于Groovy DSL的配置，提高Bean配置的灵活性。
  - Spring MVC基于Servlet 3.0 开发，并且为了方便Restful开发，引入了新的RestController注解器注解，同时还增加了一个AsyncRestTemplate支持Rest客户端的异步无阻塞请求。

Spring框架版本介绍

Neuedu

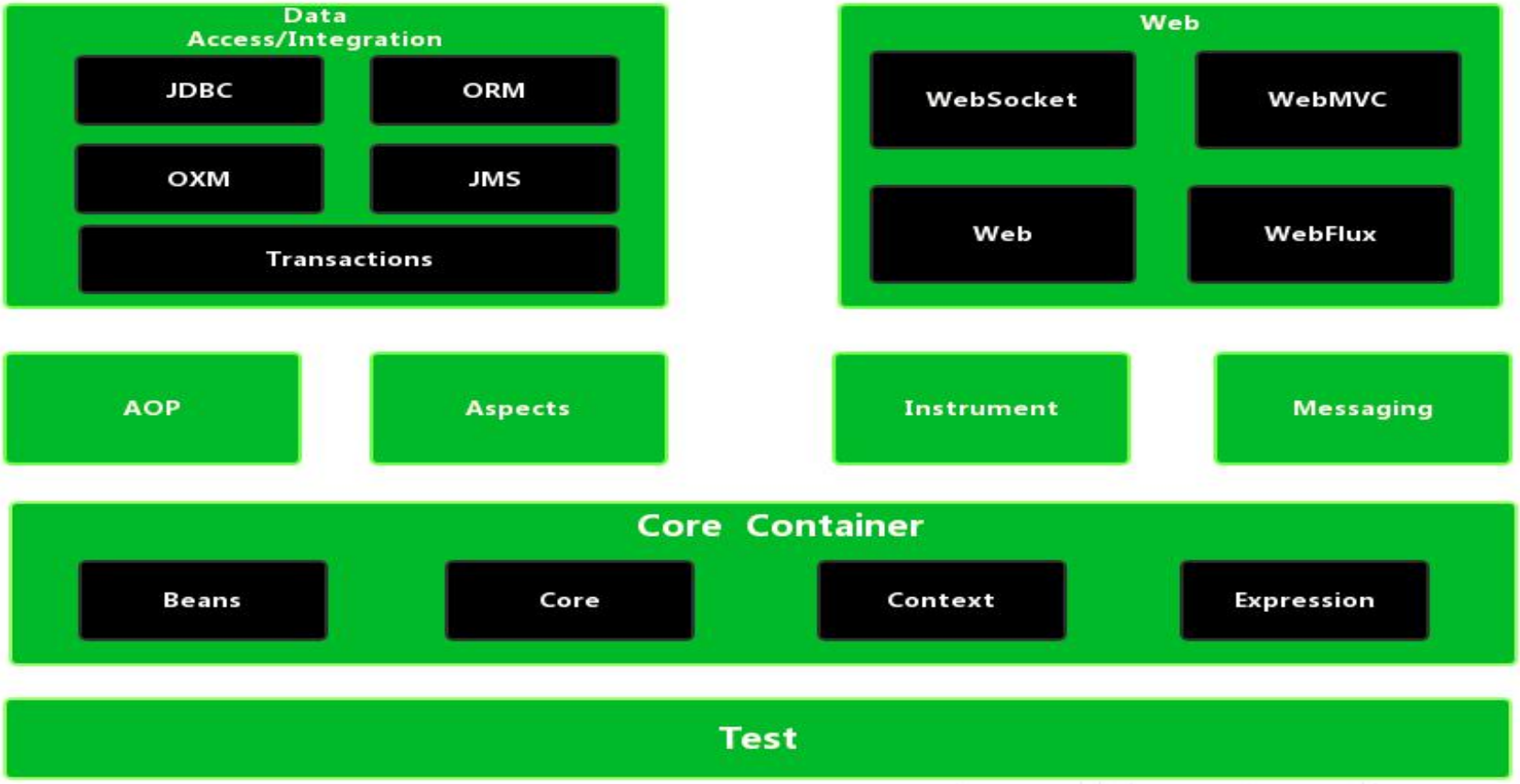
- Spring 5.x
  - Spring Framework 5.0在2017年9月28日通过Spring官方博客撰文宣告发布第一个GA版本。期间经历1年多的里程碑版本和数个RC版本，现在 5.0.0.RELEASE已正式发布到了repo.spring.io和Maven中央仓库，开发者可以通过Maven或Gradle快速获取.
  - Spring5.x的新特性：



Spring体系结构

Neuedu

- Spring 总共大约有 20 个模块，由 1300 多个不同的文件构成。而这些组件被分别整合在核心容器 （Core Container）、AOP（Aspect Oriented Programming）和设备支持（Instrmentation）、 数据访问及集成（Data Access/Integeration）、Web、报文发送（Messaging）、Test，6 个模块集合中。以下是 Spring 5 的模块结构图：



Spring体系结构

Neuedu

- 1、核心容器由spring-beans、spring-core、spring-context 和 spring-expression（Spring Expression Language, SpEL） 4 个模块组成。
  - spring-beans 和 spring-core 模块是 Spring 框架的核心模块，包含了控制反转（Inversion of Control, IOC）和依赖注入（Dependency Injection, DI）。
  - spring-context 模块构架于核心模块之上，他扩展了 BeanFactory，为她添加了 Bean 生命周期 控制、框架事件体系以及资源加载透明化等功能。此外该模块还提供了许多企业级支持，如邮件访问、 远程访问、任务调度等。
  - spring-expression 模块是统一表达式语言（EL）的扩展模块，可以查询、管理运行中的对象， 同时也方便的可以调用对象方法、操作数组、集合等。
- 2、AOP和设备支持由spring-aop、spring-aspects 和 spring-instrument 3 个模块组成。
  - spring-aop 是 Spring 的另一个核心模块，是 AOP 主要的实现模块。
  - spring-aspects 模块集成自 AspectJ 框架，主要是为 Spring AOP 提供多种AOP实现方法。
  - spring-instrument 模块是基于 JAVA SE 中的“java.lang.instrument”进行设计的，应该算是AOP 的一个支援模块，主要作用是在 JVM 启用时，生成一个代理类，程序员通过代理类在运行时修改类的字节，从而改变一个类的功能，实现 AOP 的功能。

Spring体系结构

Neuedu

- 3、数据访问及集成：由spring-jdbc、spring-tx、spring-orm、spring-jms 和 spring-oxm 5 个模块组成。
  - spring-jdbc 模块是 Spring 提供的 JDBC 抽象框架的主要实现模块，用于简化 Spring JDBC。
  - spring-tx 模块是 Spring JDBC 事务控制实现模块。
  - spring-orm 模块是 ORM 框架支持模块，主要集成 Hibernate, Java Persistence API (JPA) 和Java Data Objects (JDO) 用于资源管理、数据访问对象(DAO)的实现和事务策略。
  - spring-jms 模块（Java Messaging Service）能够发送和接受信息，自 Spring Framework 4.1 以后，他还提供了对 spring-messaging 模块的支撑。
  - spring-oxm 模块主要提供一个抽象层以支撑 OXM（OXM 是 Object-to-XML-Mapping 的缩写，它是一个 O/M-mapper，将 java 对象映射成 XML 数据，或者将 XML 数据映射成 java 对象），例如：JAXB,Castor, XMLBeans, JiBX 和 XStream 等。

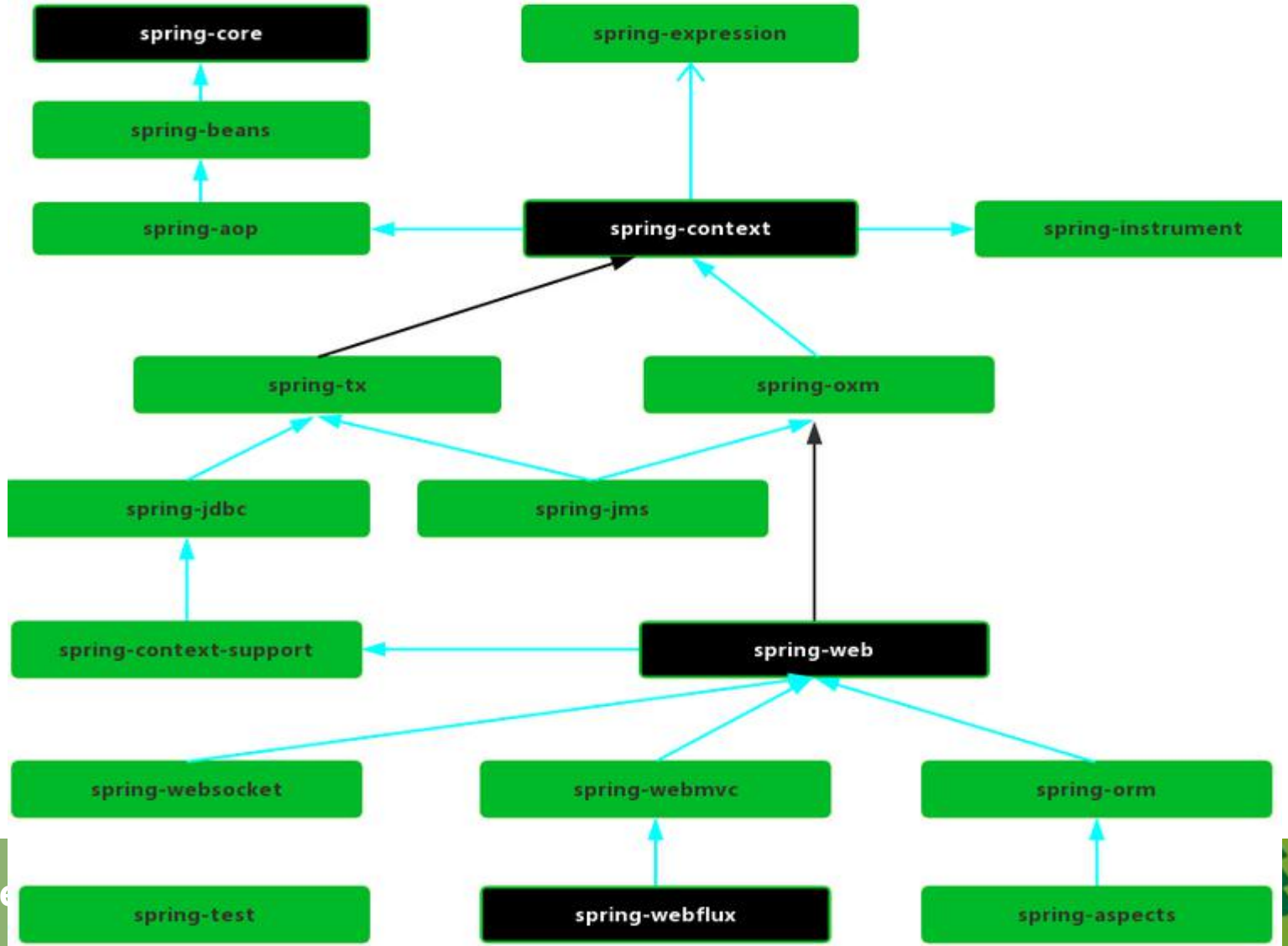
Spring体系结构

Neuedu

- 4、Web由 spring-web、spring-webmvc、spring-websocket 和 spring-webflux 4 个模块组成。
  - spring-web 模块为 Spring 提供了最基础 Web 支持，主要建立于核心容器之上，通过 Servlet 或 者 Listeners 来初始化 IOC 容器，也包含一些与 Web 相关的支持。
  - spring-webmvc模块为web应用提供了模型视图控制（MVC）和REST Web服务的实现。Spring的MVC框架可以使领域模型代码和web表单完全地分离，且可以与Spring框架的其它所有功能进行集成。
  - spring-websocket 模块为WebSocket-based 提供了支持，而且在 web 应用程序中提供了客户端和服务器端之间通信的两种方式。
  - spring-webflux 是一个新的非堵塞函数式 Reactive Web 框架，可以用来建立异步的，非阻塞，事件驱动的服务，并且扩展性非常好。
- 5、报文发送：即 spring-messaging 模块。
  - spring-messaging 是从 Spring4 开始新加入的一个模块，主要职责是为 Spring 框架集成一些基础的报文传送应用。
- 6、Test：即 spring-test 模块。
  - spring-test 模块主要为测试提供支持。

Spring体系结构

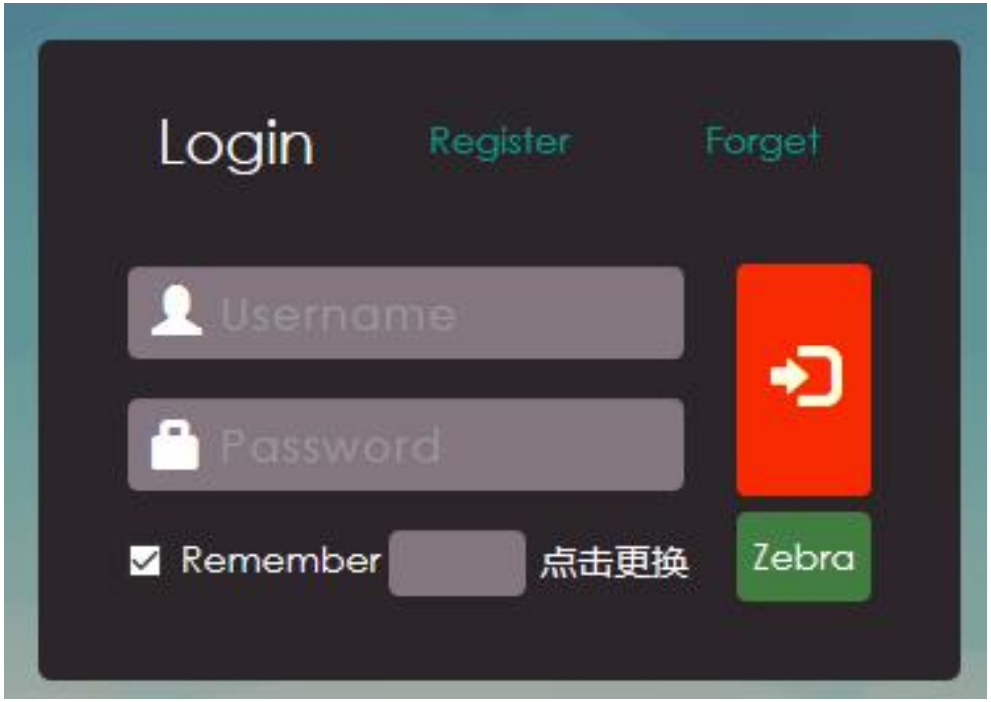
- Spirng 各模块之间的依赖关系



CONTENTS 目录

创建第一个Spring项目

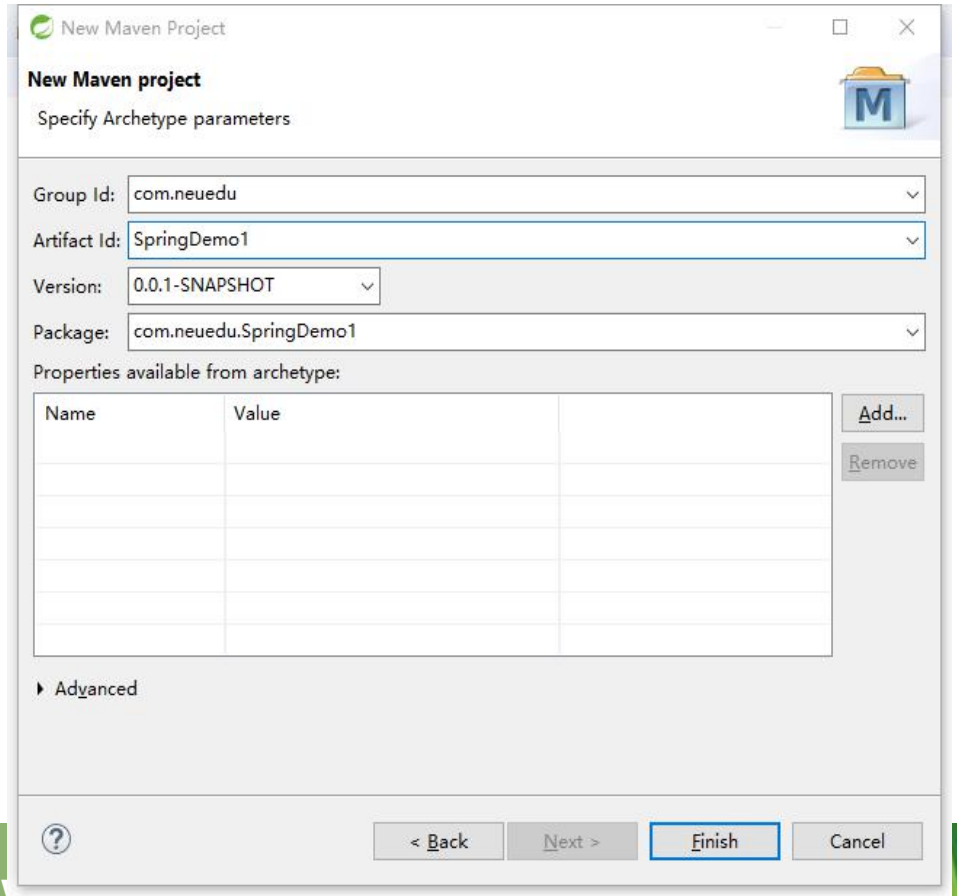
- 我们使用之前在学习Servlet时候采用的“东软跨境电商平台项目”的用户登录功能的Service层与DAO层实现为例，通过Spring来实现两者之间的依赖调用关系，讲解说明Spring项目的基本创建流程。



创建第一个Spring项目

Neuedu

- 1、使用Maven创建Java工程，并引入Spring框架类库
  - 根据实际项目需求情况，通过STS集成开发工具，创建Maven工程，并引入Spring相关jar包，本示例通过一个Java项目进行举例说明。
  - http://mvnrepository.com/ 查询依赖信息

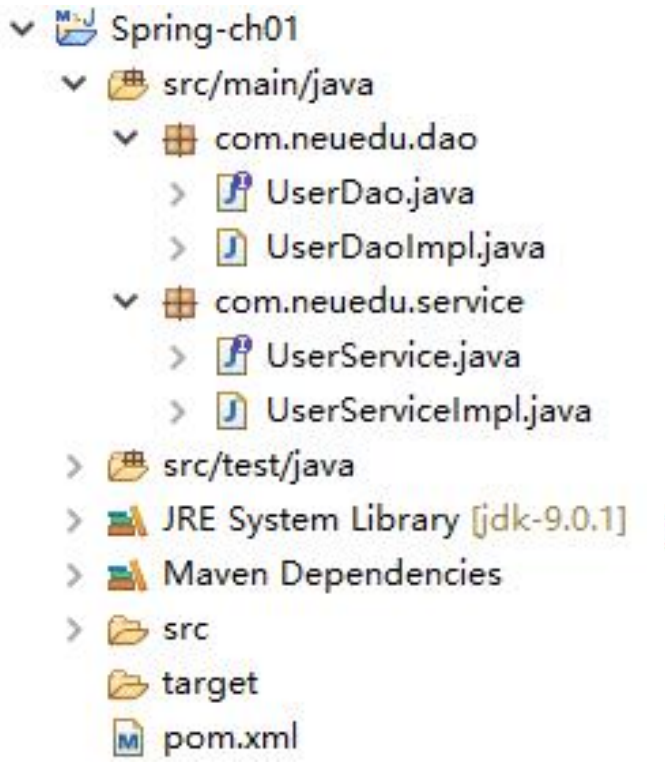



```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.0.6.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-core</artifactId>
    <version>5.0.6.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-beans</artifactId>
    <version>5.0.6.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-test</artifactId>
    <version>5.0.6.RELEASE</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
</dependency>
```

创建第一个Spring项目

Neuedu

- 2、创建Serevice、Dao接口与实现类，为了简化操作，这里的方法只需要模拟即可，不需要真实现具体代码，其中Service实现类中，我们暂时不实现对Dao接口的调用，在后面我们通过Spring提供的依赖注入来实现。




```java
public interface UserDao {
    public boolean login(String userName,String password) throws Exception;
}

public class UserDaoImpl implements UserDao {
    public boolean login(String userName, String password) throws Exception {
        System.out.println("user dao login.......");
        return false;
    }
}

public interface UserService {
    public boolean login(String userName,String password) throws Exception;
}

public class UserServiceImpl implements UserService{

    private UserDao userDao;//UserService依赖UserDao接口
    public UserServiceImpl(UserDao userDao) {
        this.userDao = userDao;
    }
    public boolean login(String userName, String password) throws Exception{
        return this.userDao.login(userName, password);
    }
}
```

Spring装配机制介绍

Neuedu

- Spring容器负责创建应用程序中的Bean，并通过DI（依赖注入）来协调对象之间的依赖关系，所以我们需要通过配置来告诉Spring需要创建哪些Bean，并且如何将他们装配在一起；

- Spring提供了几种灵活的机制去描述和装配Bean,主要有以下几种方式
  - 在Java中进行显示配置
  - 在XML中进行显示配置
  - 使用注解进行自动装配

- 本章我们先来体验使用Java和XML两种方式进行配置，后续我会详细讲解具体的配置原理及注解配置
  - 本示例中需要配置UserService、UserDao，以及两个Bean之间的依赖关系。

使用Java配置类装配Bean

Neuedu

- 使用Java代码进行装配Bean，我们需要使用两个注解来实现
  - @Configuration //该注解标明这是个配置类，该类配置说明了如何在Spring应用的上下文中进行创建和装配Bean
  - @Bean //该注解告诉Spring将这个方法返回的对象注册为Bean;
  - 对象之间的依赖关系，通过配置方法中的参数实现注入;
- 创建一个配置类AppConfig类

```
@Configuration //该注解标明这是个配置类，该类配置说明了如何在Spring应用的上下文中进行创建和装配Bean
public class AppConfig {
    @Bean(name="userDao") //该注解告诉Spring将这个方法返回的对象注册为Bean
    public UserDao getUserDao() {
        return new UserDaoImpl();
    }

    @Bean(name="userService") //当Spring创建UserService对象的时候会自动装配方法中参数类型为UserDao的对象，实现对象的依赖关系（DI）
    public UserService getUserService(UserDao userDao) {
        //UserService依赖UserDao，在装配Bean的时候，传递依赖关系，采用的是（构造器注入）
        return new UserServiceImpl(userDao);
    }
}
```

- 示例代码：Spring-ch01-java工程下AppConfig类

测试运行Java配置

Neuedu

- 现在有了bean和对应的config，我们可以创建一个运行类，对他们进行调用了：

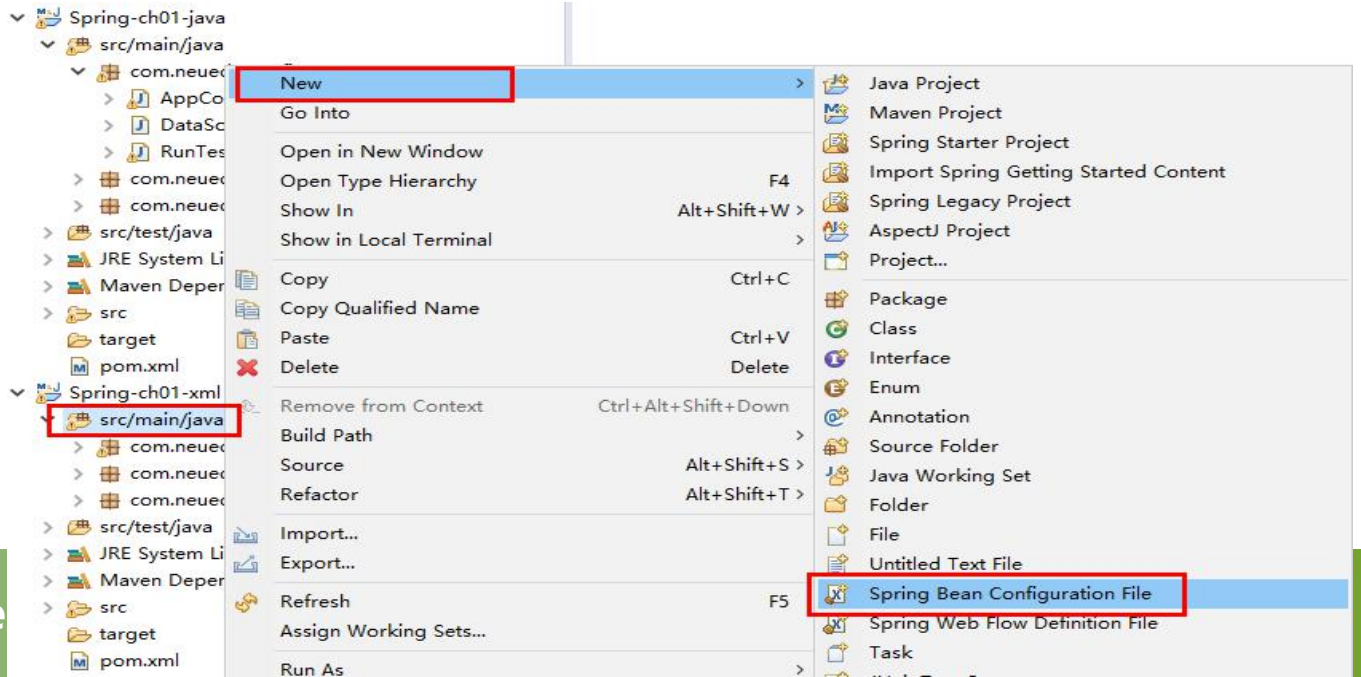
```
public class RunTest {
    //运行Main方法，进行测试
    public static void main(String[] args) throws Exception {
        //通过ApplicationContext启动一个Spring容器，并指定Spring的配置类（AppConfig）
        ApplicationContext ctx = new AnnotationConfigApplicationContext(AppConfig.class);
        //获取容器中的Bean对象
        UserService  userService = (UserService)ctx.getBean("userService");
        //调用接口方法
        userService.login("neuedu","123");

        //可以从Context中获取其他信息
        //获取Bean的数量
        System.out.println("Bean的数量" + ctx.getBeanDefinitionCount());
        //获取Bean的名称
        for (String name : ctx.getBeanDefinitionNames()) {
            System.out.println(name);
        }
    }
```

- 示例代码：Spring-ch01-java工程/RunTest.java

使用XML装配Bean

- 上面的例子我们使用的是Java配置类来进行装配Bean，在Spring2.0以前，我们都是采用的XML配置文件的方式装配Bean，对比Java配置方式，各自都有各自的优缺点，应用的场景也不同，所以大家需要掌握，后面我们对比说明不同配置方式的区别。
- 使用XML配置文件的方式装配bean，首要的就是要创建一个基于Spring配置规范的XML文件，该配置文件以<beans>为根元素（相当于Java配置的@Configuration注解），包含一个或多个<bean>元素(相当于配置类中@Bean注解)。
- 我们通过使用STS开发工具能够方便的创建Spring的xml配置文件。



使用XML装配Bean

Neuedu

- Spring配置文件的内容

AppConfig.xml

```xml
<?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">

    <!--通过类的全限定名来声明要创建的bean-->
    <bean id="userDao" class="com.neuedu.dao.UserDaoImpl"></bean>

    <!--通过类的全限定名来声明要创建的bean-->
    <bean id="userService" class="com.neuedu.service.UserServiceImpl">
        <!-- 配置依赖注入（构造器方式） -->
        <constructor-arg ref="userDao"></constructor-arg>
    </bean>
</beans>
```

- 示例代码：Spring-ch01-xml工程/AppConfig.xml

测试运行Xml配置

Neuedu

- 现在有了bean和对应的config，我们可以创建一个运行类，对他们进行调用了：

```
public class RunTest {
    //运行Main方法，进行测试
    public static void main(String[] args) throws Exception {
        //1.Java配置方式加载
            //通过ApplicationContext启动一个Spring容器，并指定Spring的配置类（AppConfig）
            //ApplicationContext ctx = new AnnotationConfigApplicationContext(AppConfig.class);
        //2.Xml配置方式加载
        ApplicationContext ctx = new ClassPathXmlApplicationContext("AppConfig.xml");

        //获取容器中的Bean对象
        UserService  userService = (UserService)ctx.getBean("userService");
        //调用接口方法
        userService.login("neuedu","123");
    }
}
```

- 示例代码：Spring-ch01-java工程/RunTest.java

CONTENTS 目录

Spring整合Junit进行单元测试

Neuedu

- 我们在后续的开发中，我们经常使用Junit进行单元测试，JUnit框架是一个开放源代码的Java测试框架，用于编写和运行可重复的测试。
- 引入依赖的jar包

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-test</artifactId>
    <version>${spring.version}</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
</dependency>
```

Spring整合Junit进行单元测试

Neuedu

- @RunWith：用于指定junit运行环境，是junit提供给其他框架测试环境接口扩展，为了便于使用spring的依赖注入，spring提供了SpringJUnit4ClassRunner作为Junit测试环境
  @ContextConfiguration,导入配置文件,支持xml、JavaConfig配置类。

```
package com.neuedu.service;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

import com.neuedu.config.AppConfig;
@RunWith(SpringJUnit4ClassRunner.class)
//@ContextConfiguration(locations= {"classpath:AppConfig.xml"})
@ContextConfiguration(classes=AppConfig.class)
public class UserServiceImplTest {

    @Autowired
    private UserService userService;
    @Test
    public void testLogin() throws Exception {
        userService.login("neuedu", "123");
    }

}
```

本章重点总结

- 如何创建一个Spring项目；
- Spring装配机制
- 使用Java和XML两种方式装配Bean
- 使用JUnit测试运行Spring项目


Neuedu