我要自学网专业课程

Spring+Spring MVC+MyBatis

SSM框架

讲师：Jane

START

章节目录

第一章

Spring的基本应用

1.1 Spring概述

Spring是当前主流的Java Web开发框架，它是为了解决企业应用开发的复杂性问题而产生的。对于一个Java开发者来说，掌握Spring框架的使用，已是其必备的技能之一。本章将对Spring框架的基础知识进行详细的讲解。

1.1.1 什么是Spring

Spring是一个开源框架，它由Rod Johnson创建。它是为了解决企业应用开发的复杂性而创建的。

Spring使用基本的JavaBean来完成以前只可能由EJB完成的事情。

然而，Spring的用途不仅限于服务器端的开发。从简单性、可测试性和松耦合的角度而言，任何Java应用都可以从Spring中受益。

- **目的**：解决企业应用开发的复杂性
- **功能**：使用基本的JavaBean代替EJB，并提供了更多的企业应用功能
- **范围**：任何Java应用

它是一个容器框架，用来装javabean（java对象），中间层框架（万能胶）可以起一个连接作用，比如说把Struts和hibernate粘合在一起运用。简单来说，Spring是一个轻量级的控制反转(IoC)和面向切面(AOP)的容器框架。

在实际开发中，通常服务器端在采用三层体系架构，分别为表示层(Web)、业务逻辑层(Service)、持久层(Dao)， Spring对每一层都提供了技术支持。

表示层 ⟹ 在表示层提供了与Struts等框架的整合

业务逻辑层 ⟹ 在业务逻辑层可以管理事务、记录日志等

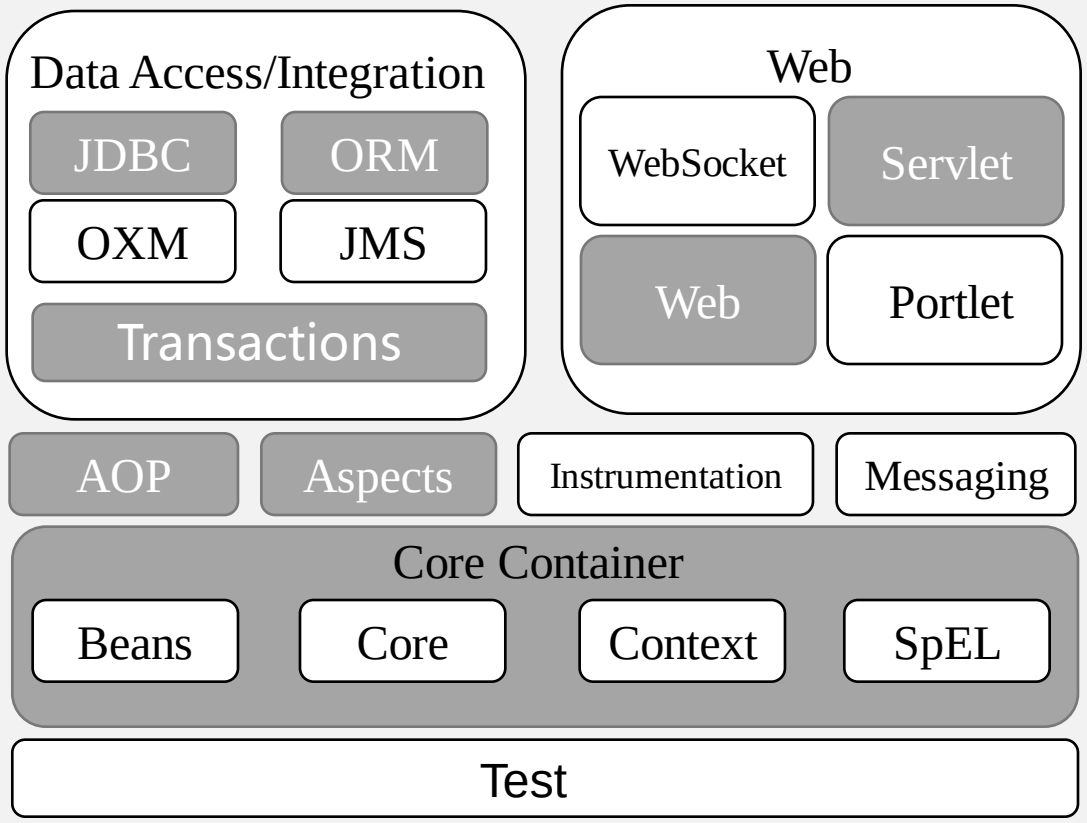
持久层 ⟹ 在持久层可以整合Hibernate、JdbcTemplate等技术

1.1.2 Spring框架的优点

- 1.使用Spring的IOC容器，将对象之间的依赖关系交给Spring，降低组件之间的耦合性，让我们更专注于应用逻辑。
- 2.可以提供众多服务，事务管理，WS等。
- 3.AOP的很好支持，方便面向切面编程。
- 4.对主流的框架提供了很好的集成支持，如hibernate,Struts2,JPA等。
- 5.Spring DI机制降低了业务对象替换的复杂性。
- 6.Spring属于低侵入，代码污染极低。
- 7.Spring的高度可开放性，并不强制依赖于Spring，开发者可以自由选择Spring部分或全部。

1.1.3 Spring的体系结构

Spring框架采用的是分层架构，它一系列的功能要素被分成20个模块。



1、核心模块（CoreContainer）

Spring核心模块包含有Core、Beans、Context和Expression Language四个小模块。其中，Core和Beans是整个Spring框架基础部分，也是Spring的核心依赖注入IoC与DI的最基本实现，Spring的其他模块大多依赖这两个功能。

- **spring-core：**其他模块的基础核心，包含Spring框架的核心工具类，Spring其他模块都要使用该包里面的类。
- **spring-beans：**Spring定义bean的支持，负责访问配置文件、创建和管理bean，支持依赖注入和控制反转的相关操作。传说中的bean工厂类就在这个jar包中。
- **spring-context：**spring运行时容器，在Core和Beans的基础上，提供对Spring的上下文支持，ApplicationContext是该包的关键，通过它，可以方便快捷的取出依赖注入的Bean。
- **spring-expression：**spring表达式语言，帮助Spring在运行时查询和操作对象。支持设置/获取对象的属性值，方法的调用。

2、AOP模块

- **spring-aop**：对于代理AOP的支持
- **spring-Aspects**：对于AspectJ的AOP支持

3、Web模块

- **spring-web**：提供基础的web功能，在Web项目中提供Spring的容器
- **spring-webmvc**：提供基于Servlet的SpringMVC
- **Spring-WebSocket**：提供WebSocket功能
- **spring-webmvc-portlet**：提供portlet的支持

4：数据库模块

- **spring-jdbc：**提供jdbc访问数据库的支持，包含了Srping对数据库访问操作进行封装的所有类，它提供了一个JDBC的抽象层，从而实现对其他厂商的支持。
- **spring-tx：**提供对事物的支持
- **spring-orm：**提供对象关系-映射的支持，使得Spring可以方便的整合其他第三方ORM库如JAP、Mybatis、Hibernate等
- **spring-oxm：**提供对象xml映射支持
- **spring-jms：**提供对java消息服务的支持

1.1.4 Spring的下载及目录结构

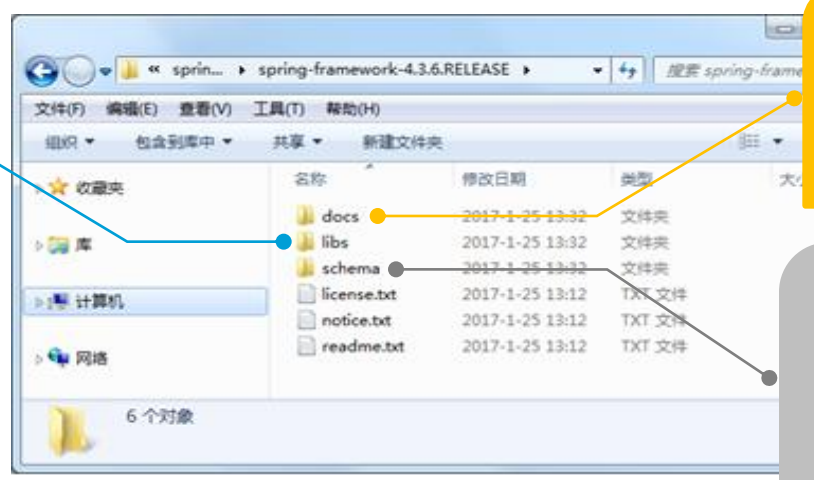
Spring开发所需的jar包分为两个部分:Spring框架包和第三方依赖包。

1. Spring框架包

下载地址： http://repo.spring.io/simple/libs-release-local/org/springframework/spring/4.3.6.RELEASE/

下载后的解压目录如下：

libs文件夹中包含JAR包和源码



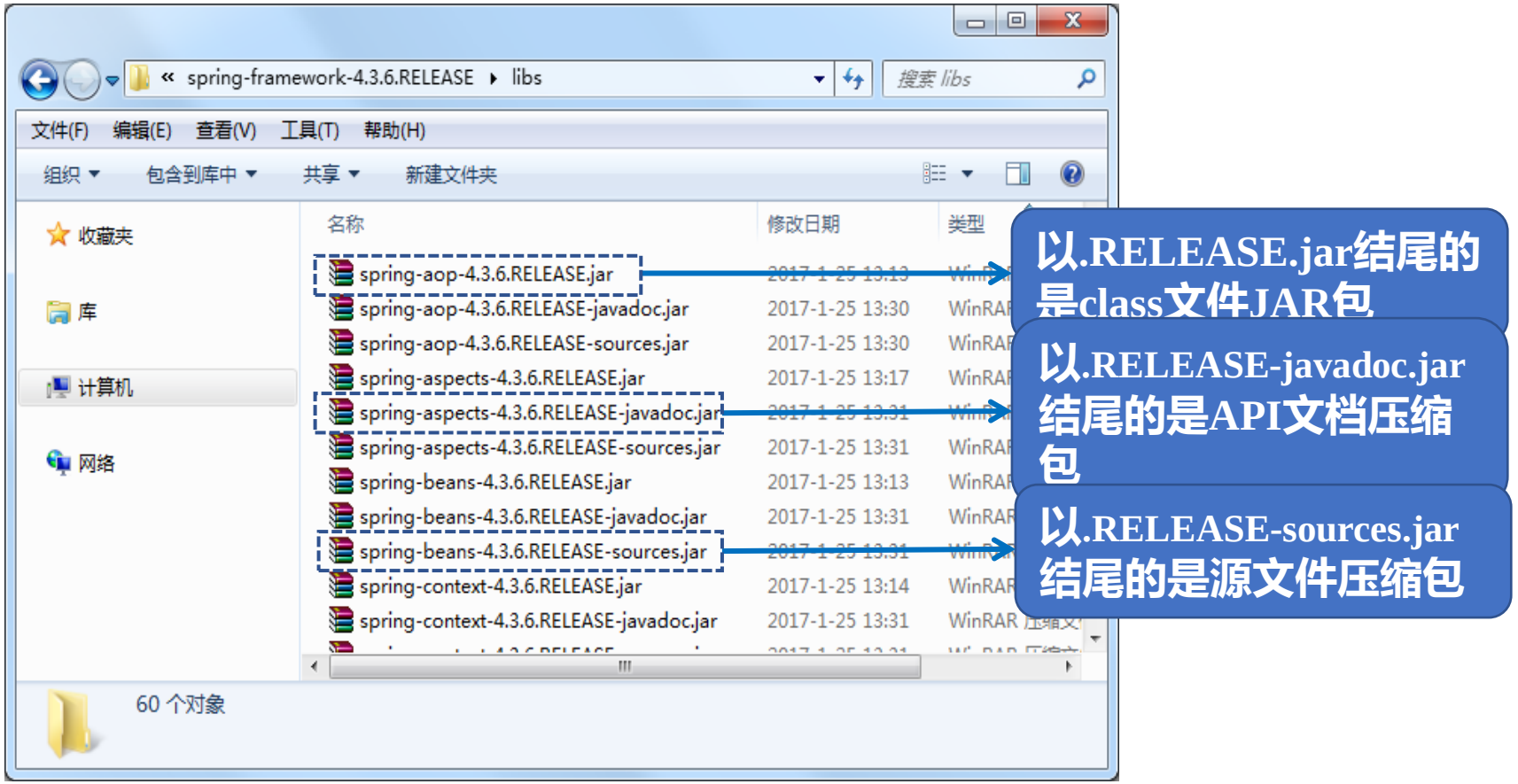
docs文件夹中包含API文档和开发规范

Schema文件夹中包含开发所需要的schema文件

小提示:

本教程开发中将使用IDEA开发工具，IDEA中，将由Maven做为Jar包管理工具，所以实际并不需要下载这些JAR包。

打开libs目录可以看到60个JAR文件，具体如下：



在libs目录中有四个Spring的基础包，分别对应Spring核心容器的四个模块。

- spring-core-4.3.6.RELEASE.jar

  包含Spring框架的核心工具类，Spring其它组件都要用到这个包里的类。

- spring-beans-4.3.6.RELEASE.jar

  所有应用都要用到的JAR包，它包含访问配置文件、创建和管理Bean以及进行控制反转或者依赖注入操作相关的所有类。

- spring-context-4.3.6.RELEASE.jar

  提供了在基础IoC功能上的扩展服务，还提供了许多企业级服务的支持

- spring-expression-4.3.6.RELEASE.jar

  定义了Spring的表达式语言。

2. 第三方依赖包

在使用Spring开发时，除了要使用自带的JAR包外，Spring的核心容器还需要依赖commons.logging的JAR包。

下载地址：http://commons.apache.org/proper/commons-logging/download_logging.cgi

1.2 Spring的核心容器

Spring容器会负责控制程序之间的关系，而不是由程序代码直接控制。Spring为我们提供了两种核心容器，分别为BeanFactory和ApplicationContext，本节将对这两种核心容器进行简单介绍。

1.2.1 BeanFactory

创建BeanFactory实例时，需要提供Spring所管理容器的详细配置信息，这些信息通常采用XML文件形式来管理，其加载配置信息的语法如下：

```
BeanFactory beanFactory =
    new XmlBeanFactory(new FileSystemResource('F: /applicationContext.xml'));
```

XML配置文件的位置

小提示：这种加载方式在实际开发中并不多用，读者作为了解即可。

1.2.2 ApplicationContext

ApplicationContext是BeanFactory的子接口，是另一种常用的Spring核心容器。它由org.springframework.context.ApplicationContext接口定义，不仅包含了BeanFactory的所有功能，还添加了对国际化、资源访问、事件传播等方面的支持。创建ApplicationContext接口实例，通常采用**两种方法**，具体如下：

1. 通过ClassPathXmlApplicationContext创建

```
ApplicationContext applicationContext =
                    new ClassPathXmlApplicationContext(String configLocation);
```

ClassPathXmlApplicationContext会从类路径classPath中寻找指定的XML配置文件，找到并装载完成ApplicationContext的实例化工作。

2. 通过FileSystemXmlApplicationContext创建

```
ApplicationContext applicationContext =
                    new FileSystemXmlApplicationContext(String configLocation);
```

FileSystemXmlApplicationContext会从指定的文件系统路径（绝对路径）中寻找指定的XML配置文件，找到并装载完成ApplicationContext的实例化工作。

在Java项目中，会通过ClassPathXmlApplicationContext类来实例化ApplicationContext容器。而在Web项目中，ApplicationContext容器的实例化工作会交由Web服务器来完成。

Web服务器实例化ApplicationContext容器时，通常会使用ContextLoaderListener来实现，此种方式只需要在web.xml中添加如下代码：

```
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>
        classpath:spring/applicationContext.xml
    </param-value>
</context-param>
<listener>
    <listener-class>
        org.springframework.web.context.ContextLoaderListener
    </listener-class>
</listener>
```

创建Spring容器后，就可以获取Spring容器中的Bean。Spring获取Bean的实例通常采用以下**两种**方法：

- Object getBean(String name);

  根据容器中Bean的id或name来获取指定的Bean，获取之后需要进行强制类型转换。

- <T> T getBean(Class<T> requiredType);

  根据类的类型来获取Bean的实例。由于此方法为泛型方法，因此在获取Bean之后不需要进行强制类型转换。

小提示

BeanFactory和ApplicationContext两种容器都是通过XML文件配置和加载Bean的。二者的主要区别在于，如果Bean的一个属性没有注入，使用BeanFactory加载后，在第一次调用getBean()方法时会抛出异常，而ApplicationContext则在初始化时自检，这样有利于检查所依赖的属性是否注入。因此，在实际开发中，通常优先使用ApplicationContext，而只有在系统资源较少时，才考虑使用BeanFactory。

1.3 Spring的入门程序

开发工具: IDEA 2018版
下载地址: https://www.jetbrains.com/idea/

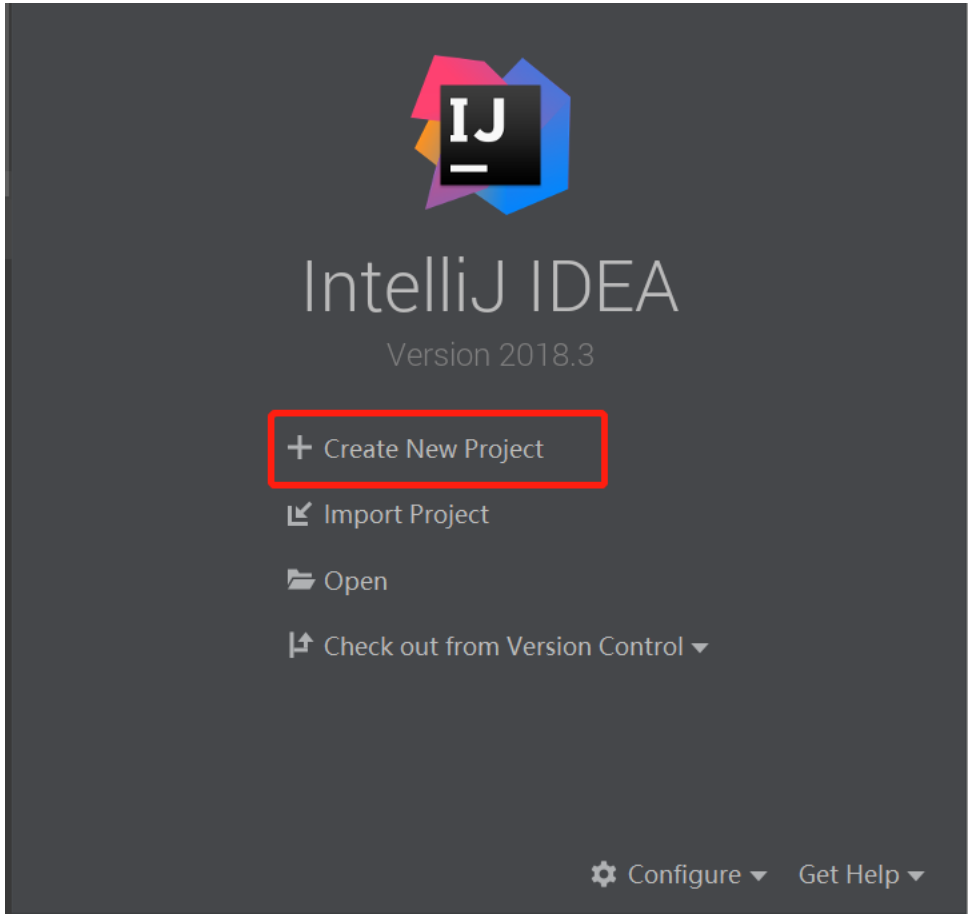
IDEA 全称 IntelliJ IDEA，是java编程语言开发的集成环境。IntelliJ在业界被公认为最好的java开发工具之一，尤其在智能代码助手、代码自动提示、重构、J2EE支持、各类版本工具(git、svn等)、JUnit、CVS整合、代码分析、 创新的GUI设计等方面的功能可以说是超常的。IDEA是JetBrains公司的产品，这家公司总部位于捷克共和国的首都布拉格，开发人员以严谨著称的东欧程序员为主。它的旗舰版本还支持HTML，CSS，PHP，MySQL，Python等。免费版只支持Java等少数语言。

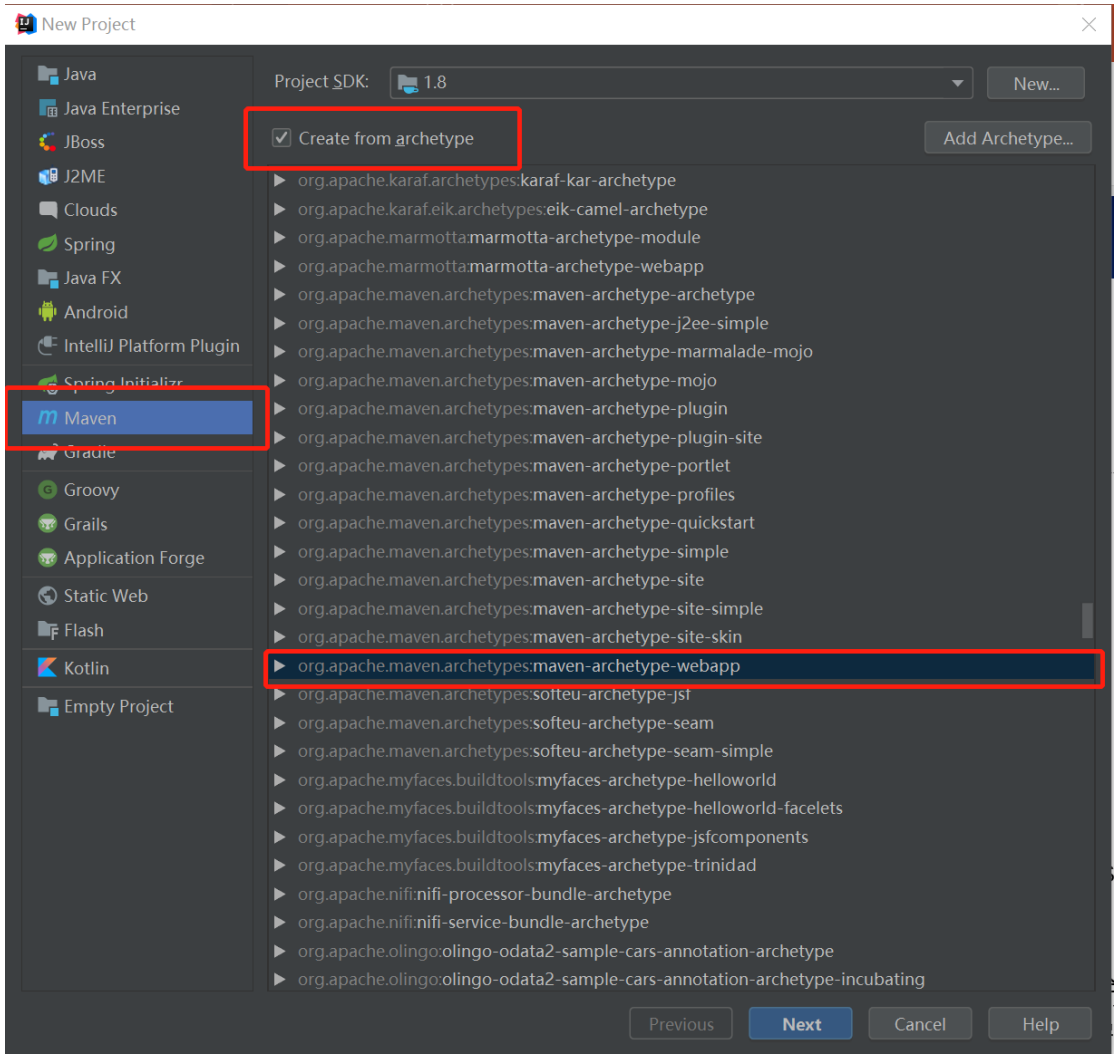
1.3 Spring的入门程序

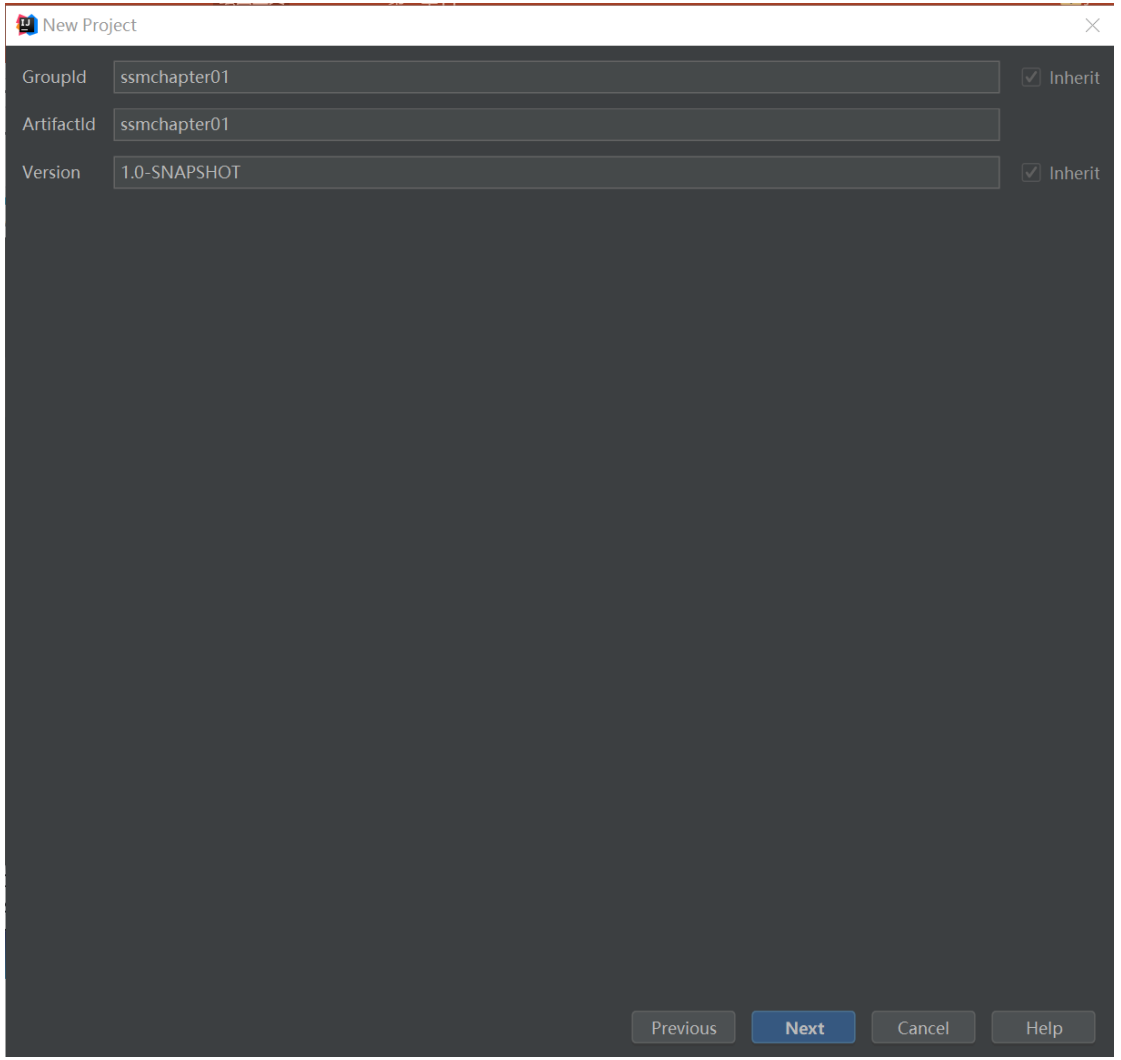
Apache Maven 是一个项目管理的工具。基于 项目对象模型POM（ project object model ）思想。 它可以帮助你管理项目的编译，文档，报告等实施过程。

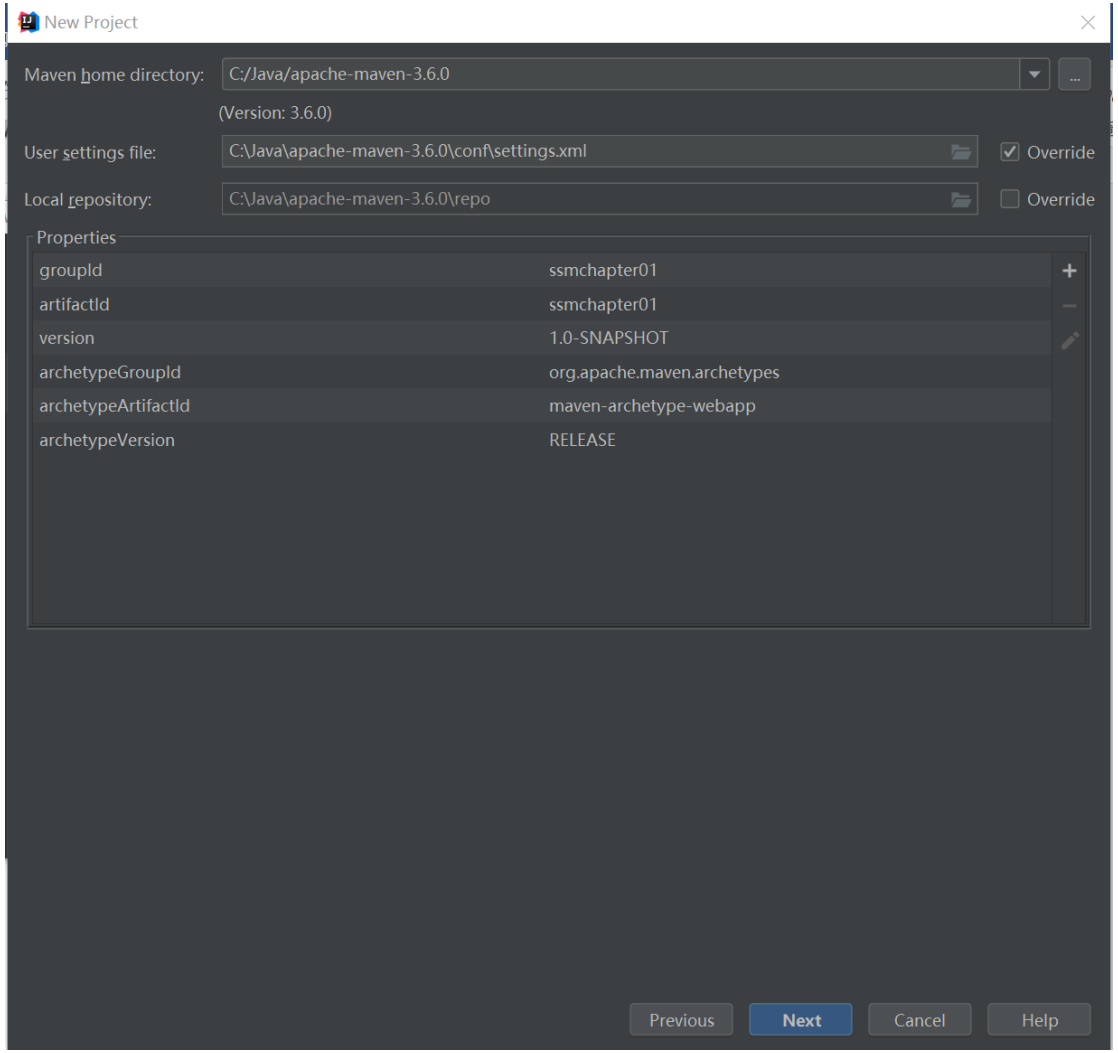
我们可以把maven当做一个jar包管理工具,以前我们的项目都是本地导入jar,而现在只需要配置pom脚本,将我们的jar包脚本到配置到pom中即可,即由本地依赖改为远程依赖,maven工具会根据脚本从公库中下载jar包,和本地依赖没有什么区别,项目发布到tomcat时idea会把项目中pom中的jar都下载并和其他文件打包成一个war包发布到tomcat下。

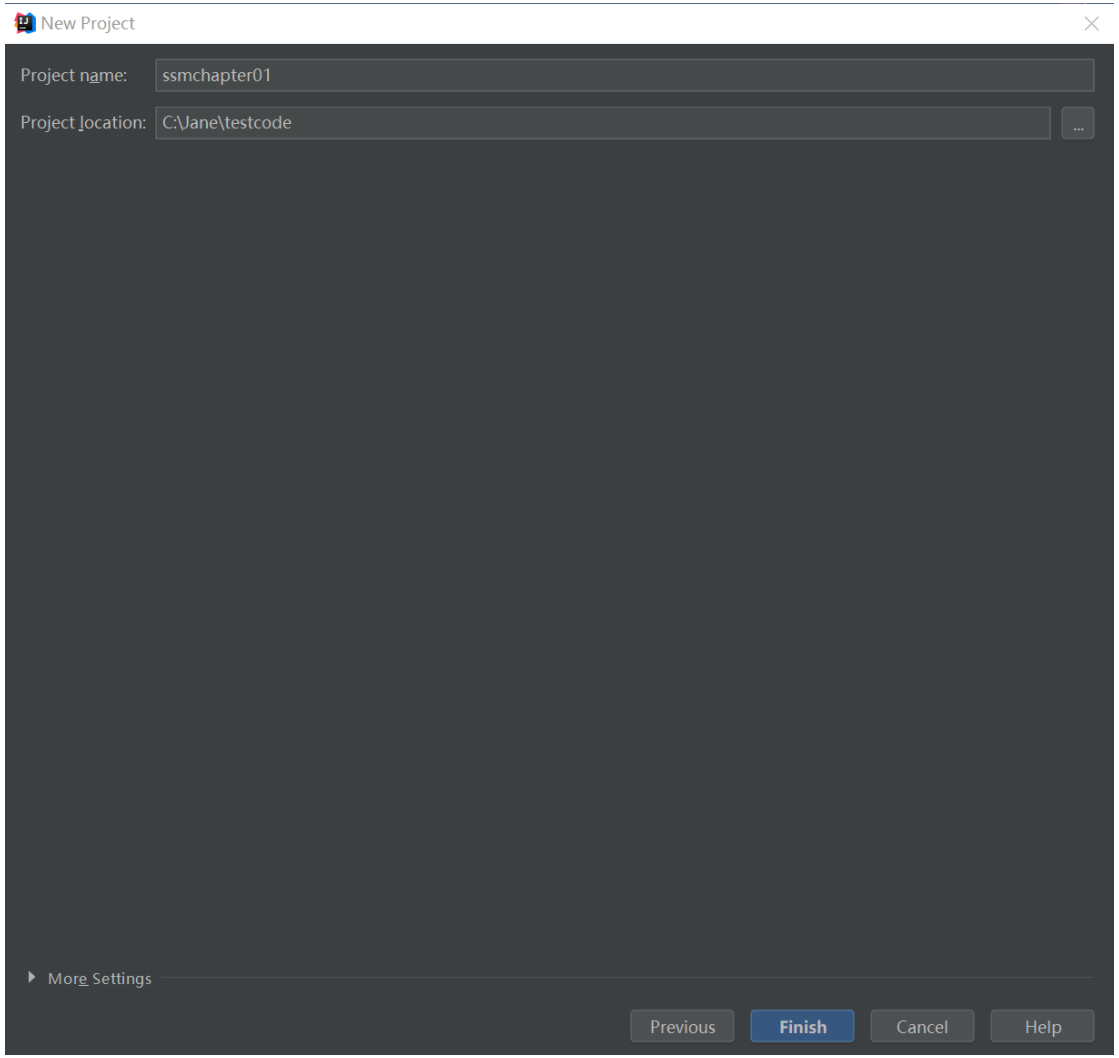
(1)在IDEA中，创建一个ssmchapter01的项目



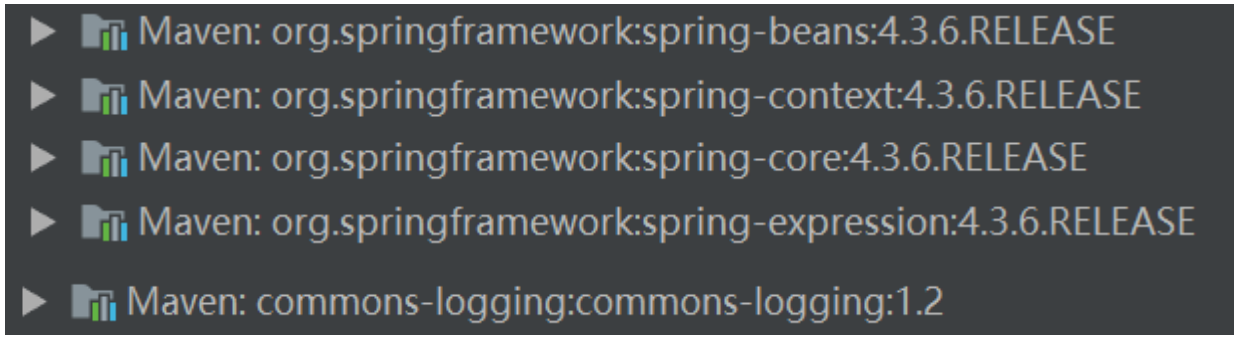








(2)在pom文件里，导入Spring的4个基础包以及commons-logging的JAR包。



https://mvnrepository.com/
输入所需要的JAR包，寻找对应的版本,复制内容，加到pom文件中。依次加入所需要的所有JAR包。

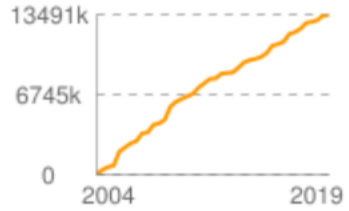
Maven Repository: commons-l

https://mvnrepository.com/artifact/commons-logging/commons-logging/1.2

MVNREPOSITORY

Search for groups, artifacts, categories | Search

Indexed Artifacts (13.4M)



Popular Categories

- Aspect Oriented
- Actor Frameworks
- Application Metrics
- Build Tools
- Bytecode Libraries
- Command Line Parsers
- Cache Implementations
- Cloud Computing
- Code Analyzers
- Collections
- Configuration Libraries

Home » commons-logging » commons-logging » 1.2

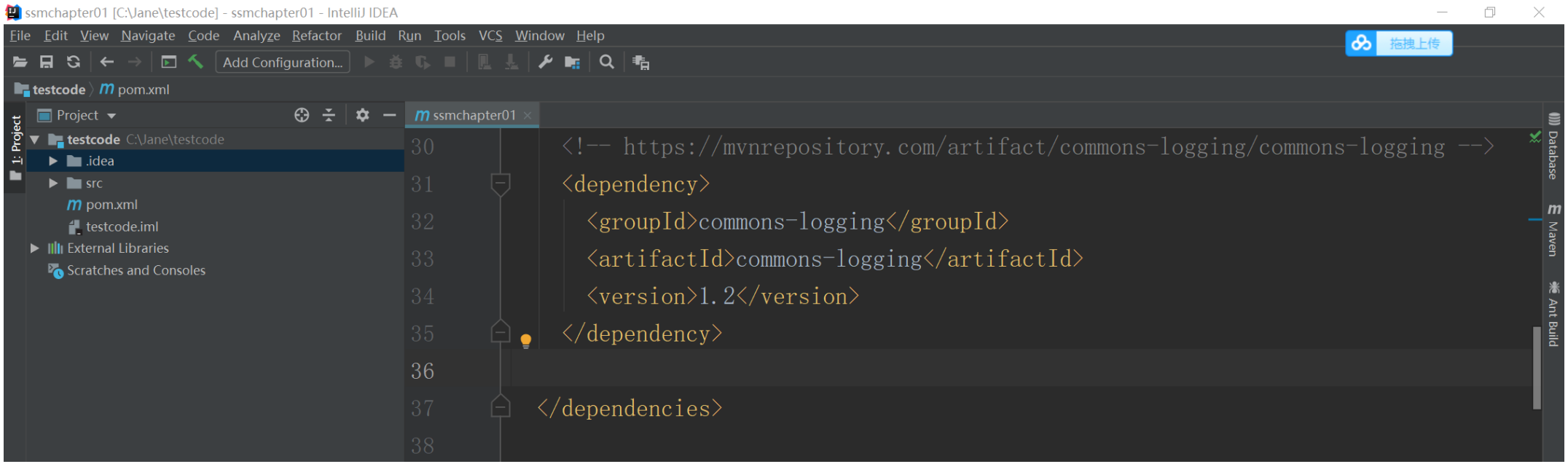
Apache Commons Logging » 1.2

Apache Commons Logging is a thin adapter allowing configurable bridging to other, well known logging systems.

License	Apache 2.0
Categories	Logging Frameworks
HomePage	http://commons.apache.org/proper/commons-logging/
Date	(Jul 05, 2014)
Files	pom (18 KB) jar (60 KB) View All
Repositories	Central Apache Releases IBiblio ImageJ Public Redhat GA
Used By	8,357 artifacts

Maven | Gradle | SBT | Ivy | Grape | Leiningen | Buildr

```
<!-- https://mvnrepository.com/artifact/commons-logging/commons-logging -->
<dependency>
    <groupId>commons-logging</groupId>
    <artifactId>commons-logging</artifactId>
    <version>1.2</version>
</dependency>
```



```
<!-- https://mvnrepository.com/artifact/commons-logging/commons-logging -->
    <dependency>
      <groupId>commons-logging</groupId>
      <artifactId>commons-logging</artifactId>
      <version>1.2</version>
    </dependency>

      <!-- https://mvnrepository.com/artifact/org.springframework/spring-beans -->
      <dependency>
          <groupId>org.springframework</groupId>
          <artifactId>spring-beans</artifactId>
          <version>4.3.6.RELEASE</version>
      </dependency>
```

```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>4.3.6.RELEASE</version>
    </dependency>

    <!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-core</artifactId>
        <version>4.3.6.RELEASE</version>
    </dependency>
```

```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-expression -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-expression</artifactId>
        <version>4.3.6.RELEASE</version>
    </dependency>
```

(3)右键点击main目录，new-Directory，新建一个Java目录，右键点击Java目录-Mark Directory as-Sources root源代码目录，这个文件夹及其子文件夹中包含的源代码，就可以编译为构建过程的一部分。

(4)在Java目录下建包，右键点击Java目录，new-Paskage，包名com.ssm.ioc。

(5)在com.ssm.ioc包下，创建接口，new-Java class，在kind下拉菜单中选择interface，接口名UserDao，在接口中定义一个say()方法。

```
package com.ssm.ioc;
public interface UserDao {
    public void say();
}
```

(6)在com.ssm.ioc包下，创建UserDao接口的实现类UserDaoImpl， new-Java class，Class名为UserDaoImpl，该类需要实现接口中的say()方法，并在方法中编写一条输出语句。

```
package com.ssm.ioc;
public class UserDaoImpl implements UserDao {
    public void say() {
        System.out.println("userDao say hello World !");
    }
}
```

(7)在main目录下， new-Directory，新建一个resources目录，右键点击resources目录-Mark as Resources root资源文件夹，用于应用程序中的资源文件（图像、各种配置XML和属性文件等）。

(8)在resources目录下，创建Spring的配置文件applicationContext.xml，并在配置文件中创建一个id为userDao的Bean.

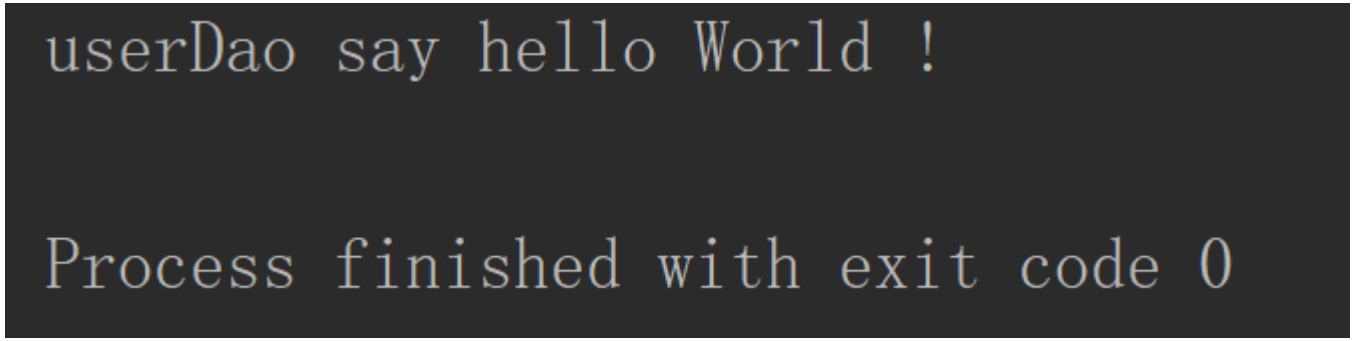
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
            http://www.springframework.org/schema/beans/spring-beans-4.3.xsd">
    <!-- 将指定类配置给Spring，让Spring创建其对象的实例 -->
    <bean id="userDao" class="com.ssm.ioc.UserDaoImpl" />
</beans>
```

(9)在com.ssm.ioc包下，创建测试类TestIoC，并在类中编写main()方法。在main()方法中，需要初始化Spring容器，并加载配置文件，然后通过Spring容器获取userDao实例（即Java对象），最后调用实例中的say()方法。

```
package com.ssm.ioc;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
```

```
public class TestIoC {
    public static void main(String[] args) {
        //1.初始化spring容器，加载配置文件
        ApplicationContext applicationContext =
                new ClassPathXmlApplicationContext("applicationContext.xml");
        //2.通过容器获取userDao实例
        UserDao userDao = (UserDao) applicationContext.getBean("userDao");
        //3.调用实例中的say()方法
        userDao.say();
    }
}
```

(10)程序执行后，控制台的结果如图。



1.4依赖注入

1.4.1依赖注入的概念

依赖注入的概念

依赖注入（Dependency Injection）简称DI，与控制反转(IoC)的含义相同，只不过这两个称呼是从两个角度描述的同一个概念。

简单来说，

没有依赖注入前，我们都是硬编码方式，new对象，然后赋值给其他对象，让他们能相互认识和协作。这里面最痛苦的就是我们需要用代码方式管理他们的生命周期。

依赖注入，提供了装配能力，框架负责new对象，以及把他们组织起来，我们唯一需要做的就是调用业务方法。

在使用Spring框架之后，对象的实例不再由调用者来创建，而是由Spring容器来创建，Spring容器会负责控制程序之间的关系，而不是由调用者的程序代码直接控制。这样，控制权由应用代码转移到了Spring容器，控制权发生了反转，这就是Spring的控制反转。

从Spring容器的角度来看，Spring容器负责将被依赖对象赋值给调用者的成员变量，这相当于为调用者注入了它依赖的实例，这就是Spring的依赖注入。

IoC（Inversion of Control，控制反转）。这是spring的核心，贯穿始终。所谓IoC，对于spring框架来说，就是由spring来负责控制对象的生命周期和对象间的关系。

Spring所倡导的开发方式就是:所有的类都会在spring容器中登记，告诉spring你是个什么东西，你需要什么东西，然后spring会在系统运行到适当的时候，把你要的东西主动给你，同时也把你交给其他需要你的东西。所有的类的创建、销毁都由 spring来控制，也就是说控制对象生存周期的不再是引用它的对象，而是spring。对于某个具体的对象而言，以前是它控制其他对象，现在是所有对象都被spring控制，所以这叫控制反转。

1.4.2依赖注入的实现方式

1.接口注入 指的就是在接口中定义要注入的信息，并通过接口完成注入。

2.set注入 指的就是在接受注入的类中定义一个Set方法，并在参数中定义需要注入的元素。

3.构造注入 构造注入指的就是接受注入的类中定义一个构造方法，并在参数中定义需要注入的元素。

下面以属性setter方法注入的方式为例，讲解一下Spring容器在应用中是如何实现依赖注入的。

(1)在com.ssm.ico包下，创建接口UserService，然后在接口中定义一个say()方法。

```
package com.ssm.ioc;
public interface UserService {
    public void say();
}
```

(2)在com.ioc包下，创建UserService接口的实现类UserServiceImpl，在类中声明userDao属性，并添加属性的setter方法。

```
package com.ssm.ioc;
public class UserServiceImpl implements UserService {
    // 声明UserDao属性
    private UserDao userDao;
    // 添加UserDao属性的setter方法，用于实现依赖注入
    public void setUserDao(UserDao userDao) {
        this.userDao = userDao;
    }
    // 实现的接口中方法
    public void say() {
        //调用userDao中的say()方法，并执行输出语句
        this.userDao.say();
        System.out.println("userService say hello World !");
    }
}
```

(3)在applicationContext.xml文件中，创建一个id为userService的Bean，该Bean用于实例化UserServiceIml类的信息，并将UserDao的实例注入到userService中。

```
<!--添加一个id为userService的实例 -->
    <bean id="userService" class="com.ssm.ioc.UserServiceImpl">
        <!-- 将id为userDao的Bean实例注入到userService实例中 -->
        <property name="userDao" ref="userDao" />
    </bean>
```

(4)在com.ssm.ioc包下，创建测试类TestDI，来进行程序测试。

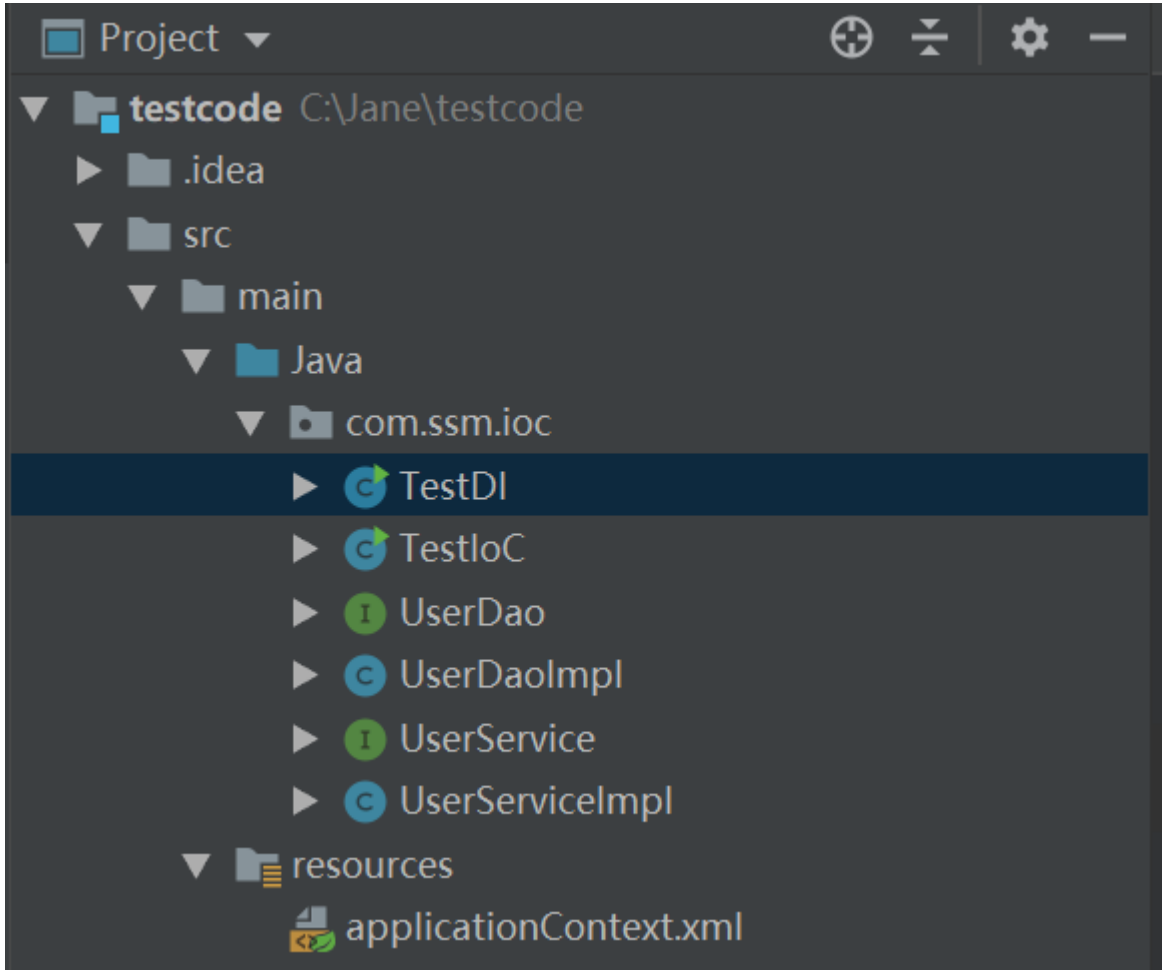
```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class TestDI {
    public static void main(String[] args) {
        //1.初始化spring容器，加载配置文件
        ApplicationContext applicationContext =
                new ClassPathXmlApplicationContext("applicationContext.xml");
        //2.通过容器获取UserService实例
        UserService userService = (UserService)
applicationContext.getBean("userService");
        //3.调用实例中的say()方法
        userService.say();
    }
```

执行程序后，控制台的输入结果如下：

```
userDao say hello World !
userService say hello World !


Process finished with exit code 0
```

本章示例的最终的目录结果如下：



本篇结束

希望大家学有所成

END