Spring前世今生与Spring编程思想

1、Spring的前世今生

相信经历过不使用框架开发Web项目的70后、80后都会有如此感触，如今的程序员开发项目太轻松了，基本只需要关心业务如何实现，通用技术问题只需要集成框架便可。早在2007年 ,一个基于Java语言的开源框架正式发布，取了一个非常有活力且美好的名字，叫做Spring，带来了Java的春天。它是一个开源的轻量级Java SE （Java标准版本）/Java EE （Java企业版本）开发应用框架，其目的是用于简化企业级应 用程序开发。应用程序是由一组相互协作的对象组成。而在传统应用程序开发中，一个完整的应用是由一组相互协作的对象组成。所以开发一个应用除了要开发业务逻辑之外，最多的是关注如何使这些对象协作来完成所需功能，而且要低耦合、高聚合。业务逻辑开发是不可避免的，那如果有个框架出来帮我们来创建对象及管理这些对象之间的依赖关系。可能有人说了,比如"抽象工厂、工厂方法模式" 不也可以帮我们创建对象，" 生成器模式" 帮我们处理对象间的依赖关系，不也能完成这些功能吗？可是这些又需要我们创建另一些工厂类、生成器类，我们又要而外管理这些类，增加了我们的负担，如果能有种通过配置方式来创建对象，管理对象之间依赖关系,我们不需要通过工厂和生成器来创建及管理对象之间的依赖关系，这样我们是不是减少了许多工作，加速了开发，能节省出很多时间来干其他事。Spring 框架刚出来时主要就是来完成这个功能。

Spring框架除了帮我们管理对象及其依赖关系，还提供像通用日志记录、性能统计、安全控制、异 常处理等面向切面的能力，还能帮我管理最头疼的数据库事务，本身提供了一套简单的JDBC访问实现,提供与第三方数据访问框架集成（如 Hibernate. JPA，Mybatis）,与各种Java EE技术整合（如 Java Mail、任务调度等等），提供一套自己的Web层框架Spring MVC、而且还能非常简单的与第三方Web框架集 成。从这里我们可以认为Spring是一个超级粘合大平台，除了自己提供功能外，还提供粘合其他技术 和框架的能力，从而使我们可以更自由的选择到底使用什么技术进行开发。而且不管是JAVASE （C/S架构）应用程序还是JAVA EE （ B/S架构）应用程序都可以使用这个平台进行开发。如今的Spring已经不再是一个框架，早已成为了一种生态。SpringBoot的便捷式开发实现了零配置，SpringCloud全 家桶，提供了非常方便的解决方案。接下来，让我们来深入探讨Spring到底能给我们带来什么?

2、Spring编程思想

2.1 —切从Bean开始

说到Bean这个概念，还得从Java的起源说起。早在1996年 ， Java还只是一个新兴的、初出茅庐 的编程语言。人们之所以关注她仅仅是因为，可以使用Java的Applet来开发Web应用，作为浏览器 组件。但开发者们很快就发现这个新兴的语言还能做更多的事情。与之前的所有语言不同，Java让模块 化构建复杂的系统成为可能（当时的软件行业虽然在业务上突飞猛进，但当时开发用的是传统的面向过程开发思想，软件的开发效率一直踟蹰不前。伴随着业务复杂性的不断加深，开发也变得越发困难。其实 ， 当时也是OOP思想飞速发展的时期，她在80年代末被提出，成熟于90年代,现今大多数编程语言都已经是面向对象的）。

同年12月 ， Sun公司发布了当时还名不见经传但后来人尽皆知的JavaBean 1.00-A规范。早期的 Java Bean规范针对于Java ,她定义了软件组件模型。这个规范规定了一整套编码策略，使简单的Java 对象不仅可以被重用，而且还可以轻松地构建更为复杂的应用。尽管JavaBean最初是为重用应用组件 而设计的，但当时他们却是主要用作构建窗体控件，毕竟在PC时代那才是主流。但相比于当时正如日中天的Delphi. VB和 C++ , 它看起来还是太简易了，以至于无法胜任任何"实际的"工作需要。

复杂的应用通常需要事务、安全、分布式等服务的支持，但 JavaBean并未直接提供。所以到了 1998 年 3 月， Sun公司发布了 EJB 1.0规范,该规范把Java组件的设计理念延伸到了服务器端，并提供了 许多必须的企业级服务，但他也不再像早期的JavaBean那么简单了。实际上,除了名字叫EJB Bean 以外，其他的和JavaBean关系不大了。

尽管现实中有很多系统是基于EJB构建的，但 EJB从来没有实现它最初的设想：简化开发。EJB的声明式编程模型的确简化了很多基础架构层面的开发，例如事务和安全；但另一方面EJB在部署描述符和配套代码实现等方面变得异常复杂。随看时间的推移，很多开发者对EJB已经不再抱有幻想，开始寻求更简洁的方法。

现在Java组件开发理念重新回归正轨。新的编程技术AOP和 DI的不断出现，他们为JavaBean提 供了之前EJB才能拥有的强大功能。这些技术为POJO提供了类似EJB的声明式编程模型，而没有引入任何EJB的复杂性。当简单的JavaBean足以胜任时，人们便不愿编写笨重的EJB组件了。

客观地讲，EJB的发展甚至促进了基于POJO的编程模型。引入新的理念，最新的EJB规范相比之前的规范有了前所未有的简化，但对很多开发者而言，这一切的一切都来得太迟了。到了 EJB3规范发布时，其他基于POJO的开发架构已经成为事实的标准了，而 Spring框架也就是在这样的大环境下出现的。

换个形象的说法：

EJB生不逢时！

EJB最初的设计思想考虑的是为分布式的应用服务的，分布式是针对大型应用构造的跨平台的协作计算，EJB最初的目的就是为这种计算服务的。但是软件发展到目前为止，大多数应用不需要采用分布式的解决方案，因此用EJB显得太臃肿了。Spring的出现恰恰为了解决这个问题。举个例子来说，EJB就是导弹，专门设计为打高空飞机。但是现在发现飞机不多。于是将它用来对付步兵，这个实在太糟糕了。这个时候有人发明了狙击步枪（Spring），发现对付步兵太好用了。

同时，因为Spring太好用了，所以后来很多人围绕狙击步枪做了大量的配套工作，使得Spring这个狙击步枪具有了高射炮的功能（分布式Session，分布式事务，分布式锁等），也可以打飞机了！

这个时候EJB3出现的已经太晚了，虽然它简化了太多的开发，但是，行业已经形成，Spring已经一统天下！

2.2 Spring的设计初衷

Spring是为解决企业级应用开发的复杂性而设计,她可以做很多事。但归根到底支撑Spring的仅 仅是少许的基本理念，而所有的这些基本理念都能可以追溯到一个最根本的使命：简化开发。这是一个郑重的承诺，其实许多框架都声称在某些方面做了简化。而 Spring则立志于全方面的简化Java开发。
对此,她主要采取了 4 个关键策略:

1、基于POJO的轻量级和最小侵入性编程;

2、通过依赖注入和面向接口松耦合；

3、基于切面和惯性进行声明式编程；

4、通过切面和模板减少样板式代码；

而他主要是通过：面向Bean(BOP).依赖注入(DI) 以及面向切面(AOP) 这三种方式来达成的。

2.3 BOP编程伊始

Spring 是面向 Bean 的编程(Bean Oriented Programming, BOP) , Bean 在 Spring 中才是真 正的主角。Bean在Spring中作用就像Object对OOP的意义一样Spring中没有Bean也就没有Spring存在的意义。Spring提供了 IOC容器通过配置文件或者注解的方式来管理对象之间的依赖关系。

控制反转(其中最常见的实现方式叫做依赖注入(Dependency Injection , DI) , 还有一种方式叫〃依赖查找〃(Dependency Lookup , DL) ,她在 C+ + 、Java. PHP以及.NET中都运用。在最早的Spring中是包含有依赖注入方法和依赖查询的/但因为依赖查询使用频率过低，不久就被Spring移除 了 ,所以在Spring中控制反转也被直接称作依赖注入)，她的基本概念是:不创建对象，但是描述创建 它们的方式。在代码中不直接与对象和服务连接，但在配置文件中描述哪一个组件需要哪一项服务。容器 (在 Spring框架中是IOC容器)负责将这些联系在一起。

在典型的IOC场景中,容器创建了所有对象,并设置必要的属性将它们连接在一起，决定什么时间调用方法。

2.4 依赖注入的基本概念

Spring 设计的核心 org.springframework.beans 包 （架构核心是 org.springframework.core包 ），它的设计目标是与JavaBean组件一起使用。这个包通常不是由用户直接使用，而是由服务器将 其用作其他多数功能的底层中介，一个最高级抽象是BeanFactory接口 ,它是工厂设计模式的实现, 允许通过名称创建和检索象。BeanFactory也可以管理对象之间的关系。

BeanFactory最底层支持两个对象模型。

1.单例：提供了具有特定名称的全局共享实例对象，可以在查询时对其进行检索。Singleton是默认的也是最常用的对象模型。

2 .多例（原型）：确保每次检索都会创建单独的实例对象。在每个用户都需要自己的对象时，采用原型模式。

Bean工厂的概念是Spring作为IoC容器的基础。I0C则将处理事情的责任从应用程序代码转移到框架。

2.5 AOP编程理念

面向切面编程，即AOP ,是一种编程思想，它允许程序员对横切关注点或横切典型的职责分界线的行为（例如日志和事务管理）进行模块化。AOP的核心构造是方面（切面），它将那些影响多个类的行为封装到可重用的模块中。

AOP和 IOC是补充性的技术，它们都运用模块化方式解决企业应用程序开发中的复杂问题。在典型的面向对象开发方式中可能要将日志记录语句放在所有方法和Java类中才能实现日志功能。在 AOP 方式中，可以反过来将日志服务模块化，并以声明的方式将它们应用到需要日志的组件上。当然，优势就是Java类不需要知道日志服务的存在，也不需要考虑相关的代码。所以，用 Spring AOP编写的应用程序代码是松散耦合的。

AOP的功能完全集成到了 Spring事务管理、日志和其他各种特性的上下文中。

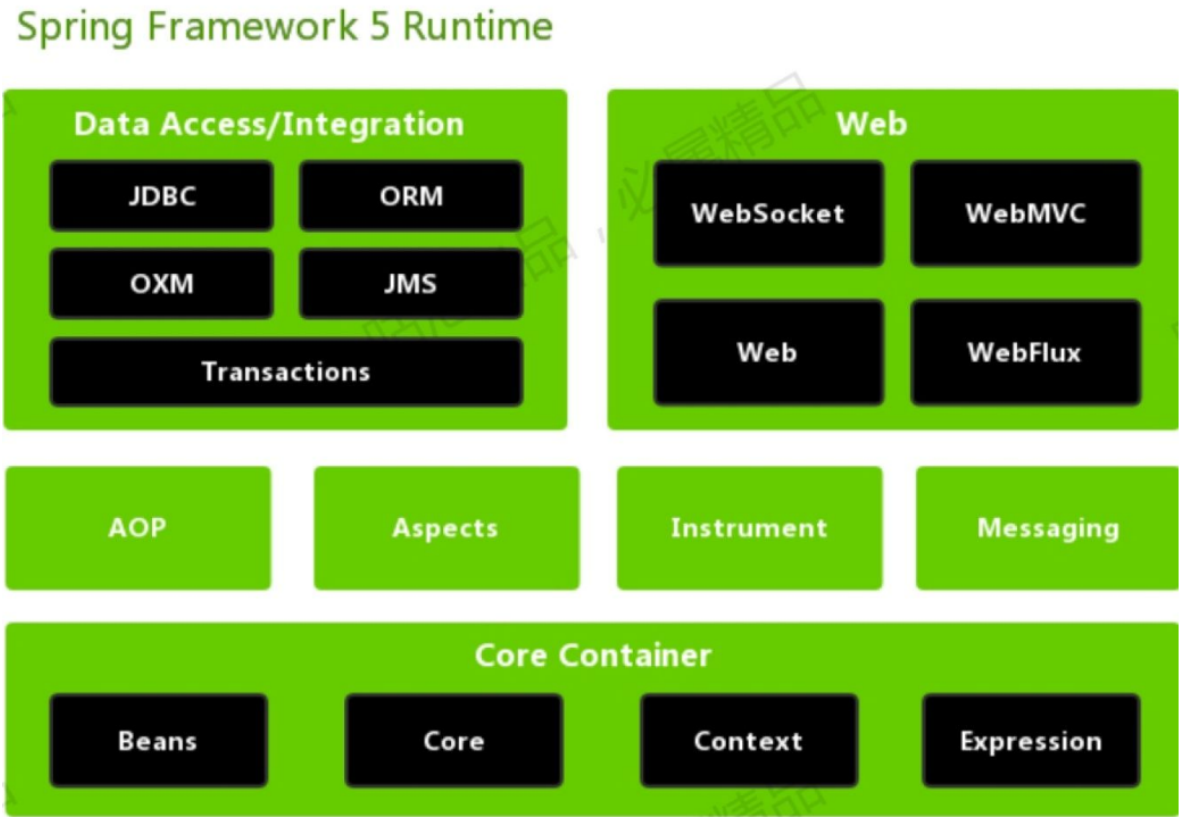
AOP编程的常用场景有:Authentication权限认证）、Auto Caching（自动缓存处理）、Error Handling（统一错误处理）、Debugging （调试信息输出）、Logging （日志记录）、Transactions （事务处理）
等。

2.6 Spring 编程思想总结

Spring思想	应用场景（特点）	一句话归纳
OOP	Object Oriented Programming （面向对象编程）	用程序归纳总结生活中一切事物。 封装、继承、多态。
BOP	Bean Oriented Programming （面向 Bean 编程）面向Bean （普通的Java类）设计程序， 解放程序员。	一切从Bean开始。
AOP	Aspect Oriented Programming（面向切面编程）找出多个类中有一定规律的代码,开发时拆开 ， 运行时再合并。 面向切面编程，即面向规则编程。	解耦，专人做专事。
IOC	Inversion of Control （控制反转） 将 new对象的动作交给Spring管 理 ， 并由 Spring保存已创建的对象（IOC容器）。	转交控制权（即控制权反转）
DI/DL	Dependency Injection （依 赖 注 入 ）或者 Dependency Lookup （依赖查找） 依赖注入、依赖查找,Spring不仅保存自己创 建的对象，而且保存对象与对象之间的关系。 注入即赋值,主要三种方式构造方法、set方法、直接赋值。	赋值

3、Springs系统架构

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



3.1 核心容器

由 spring-beans、spring-core, spring-context , spring-expression (Spring Expression Language, SpEL) 4 个模块组成。

spring-core和 spring-beans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) 。 Bean Factory 接口是 Spring 框架中的核 心接口，它是工厂模式的具体实现。Bean Factory使用控制反转对应用程序的配置和依赖性规范与实际 的应用程序代码进行了分离。但 BeanFactory容器实例化后并不会自动实例化Bean,只有当Bean被 使用时

BeanFactory容器才会对该Bean进行实例化与依赖关系的装配（多例）。

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，Applicationcontext是该模块的核心接口，她的超类是BeanFactory。与 BeanFactory不同，Applicationcontext容器实例化后会自动对所有的单实例Bean进行实例化与依 赖关系的装配，使之处于待用状态。（单例）

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。它的语法类似于传统EL ,但提供了额外的功能，最 出色的要数函数调用和简单字符串的模板函数。这种语言的特性是基 于Spring产品的需求而设计，他 可以非常方便地同Spring IOC进行交互。

3.2 AOP和设备支持

spring-aop、spring-aspects 模块

spring-aop是 Spring的另一个核心模块，是 AOP主要的实现模块。作为继OOP后 ，对程序员影 响最大的编程思想之一，AOP极大地开拓了人们对于编程的思路。在 Spring中 ，他是以JVM的动态 代理技术为基础，然后设计出了一系列的AOP横切实现，比如前置通知、返回通知、异常通知等，同 时 ,Pointcut接口来匹配切入点，可以使用现有的切入点来设计横切面，也可以扩展相关方法根据需求 进行切入。

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，但是现在基本上不用，都会使用第三方集成框架aspectjweaver。

3.3 数据访问与集成

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操 作 。 主要是提供JDBC模板方式、关系数据库对象化方式、SimpleJdbc方式、事务管理来简化JDBC编程, 主要实现类是JdbcTemplate、SimpleJdbcTemplate 以及 NamedParameterJdbcTemplate

spring-tx模块是Spring JDBC事务控制实现模块。使用Spring框 架 ,它对事务做了很好的封装, 通过它的AOP配置,可以灵活的配置在任何一层;在很多的需求和应用中，直接使用JDBC事务控 制还是有其优势的。其实，事务是以业务逻辑为基 础的；一个完整的业务应该对应业务层里的一个方法； 如果业务操作失败'则整个事务回滚;所以事务控制是绝对应该放在业务层的；但是,持久层的设计 则应该遵循一个很重要的原则：保证操作的原子性，即持久层里的每个方法都应该是不可以分割的。所以 ，在使用Spring JDBC事务控制时，应该注意其特殊性。

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-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-jms模 块 (Java Messaging Service) 能够发送和接收信息, 自 Spring Framework 4.1 以 后 ， 他还提供了对spring-messaging模块的支撑。

3.4 Web组件

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1=1 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模块众所周知是一个的Web-Servlet模 块 ,实现了 Spring MVC (model-view-Controller) 的 Web 应用。

spring-websocket模块主要是与Web前端的全双工通讯的协议。

spring-webflux是一个新的异步非堵塞函数式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。

3.5 通信报文

即 spring-messaging模 块 ,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。

3.6 集成测试

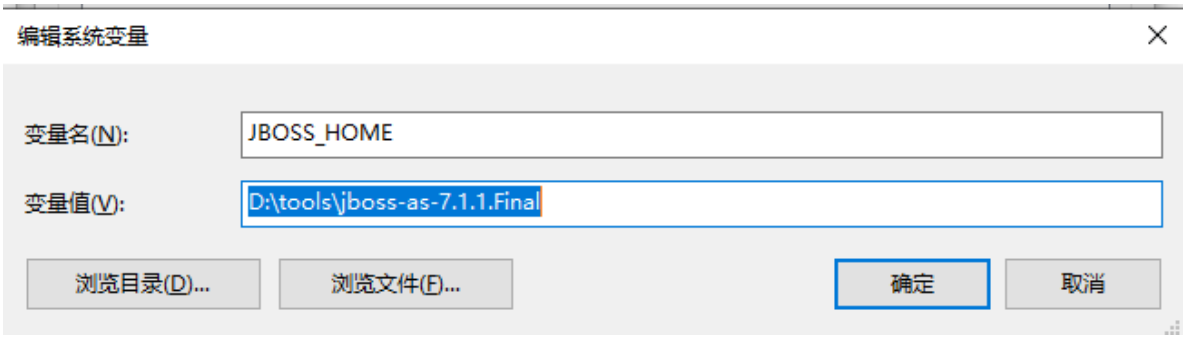
即 spring-test模块,主要为测试提供支持的，毕竟在不需要发布(程序)到你的应用服务器或者连接 到其他企业设施的情况下能够执行 些集成测试或者其他测试对于任何企业都是非常重要的。

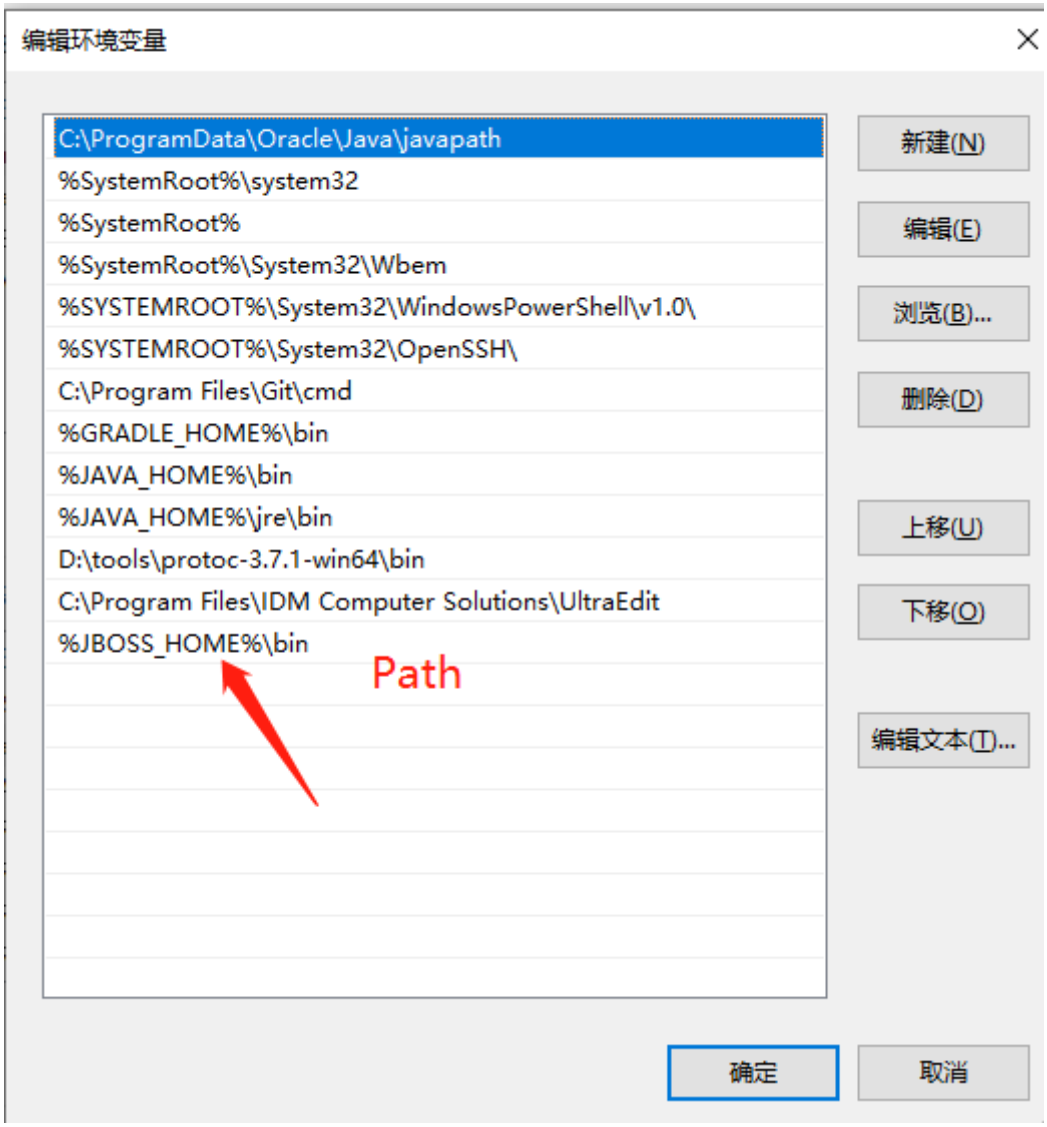
4、附录-IDEA搭建EJB入门Demo

4.1 安装JBoss

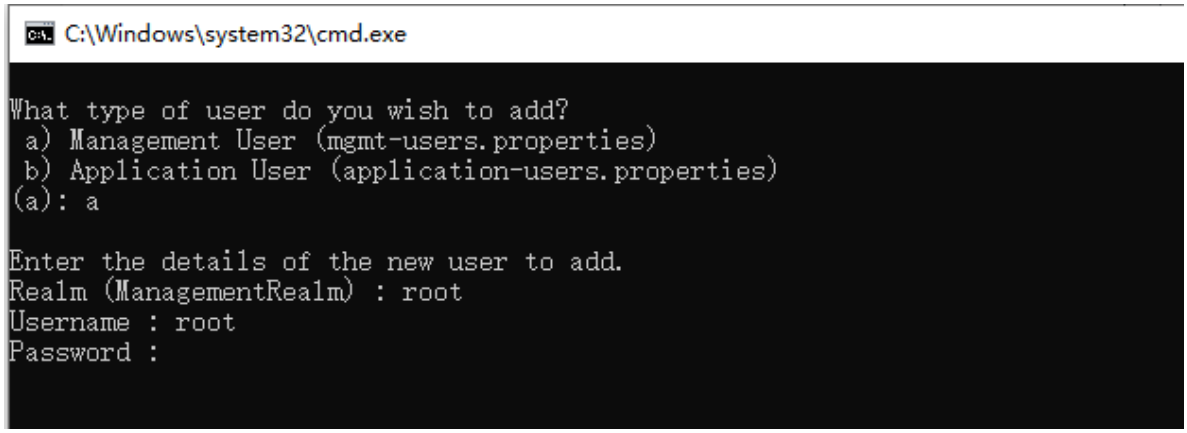
下载地址：http://jbossas.jboss.org/downloads

下载完后解压，配置环境变量，跟Java一毛一样





配置完成后需要添加用户，点击JBoss安装目录下bin目录下找到add-user.bat文件，点击运行如下图，慢慢填（账号密码不能一样）



当前7.1.1版本的JBoss只能支持JDK7及以下版本，所以你需要在本地弄一个JDK7，然后修改jboss-as-7.1.1.Final\bin\standalone.bat下JAVA_HOME

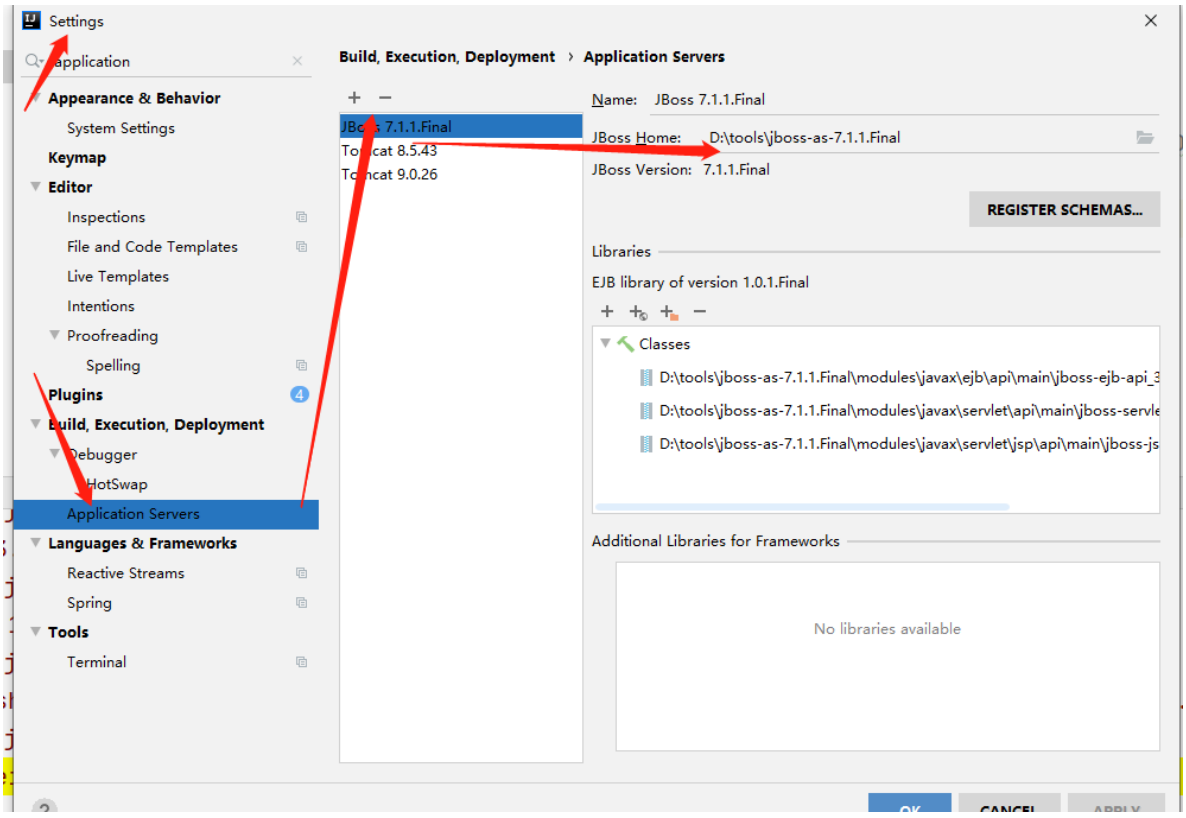
指定JDK7所在目录，这个不影响你环境变量中JDK其他版本的使用，仅仅只是为了让JBoss跑起来

```
rem Setup JBoss specific properties
set JAVA_OPTS=-Dprogram.name=%PROGNAME% %JAVA_OPTS%

if "x%JAVA_HOME%" == "x" (
  set  JAVA=java
  echo JAVA_HOME is not set. Unexpected results may occur.
  echo Set JAVA_HOME to the directory of your local JDK to avoid this message.
) else (
  set "JAVA=D:\tools\jdk1.7.0-64\bin\java"
)
```

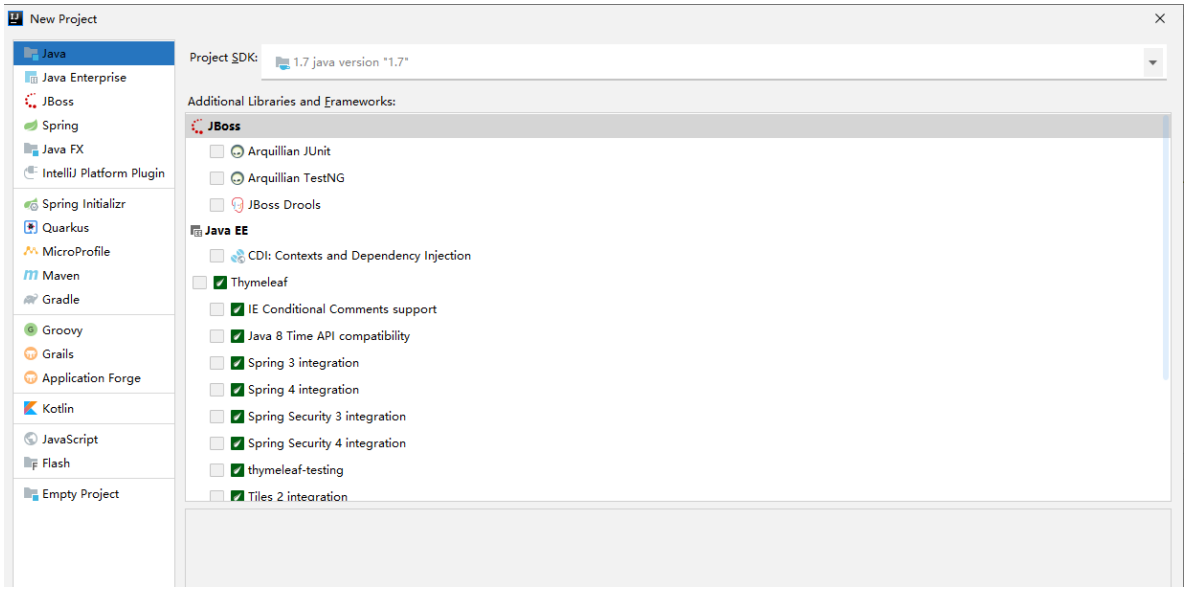
运行standalone.bat或者standalone.sh启动，http://localhost:8080启动能能进入JBoss控制台即可！

4.2 IDEA集成配置全局JBoss服务

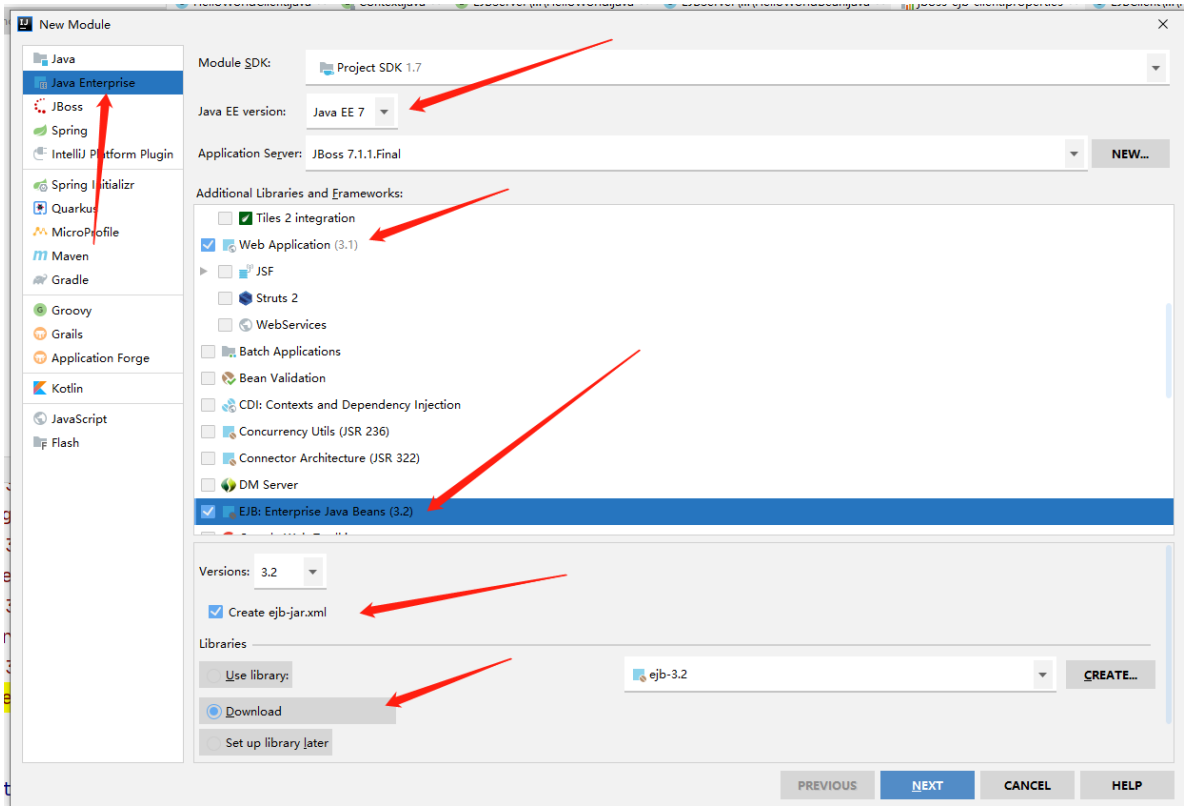


4.3 创建EJB工程（EJBServer）

创建Project，类型为简单的Java工程



右键该项目，创建Module，作为EJBServer，如图：

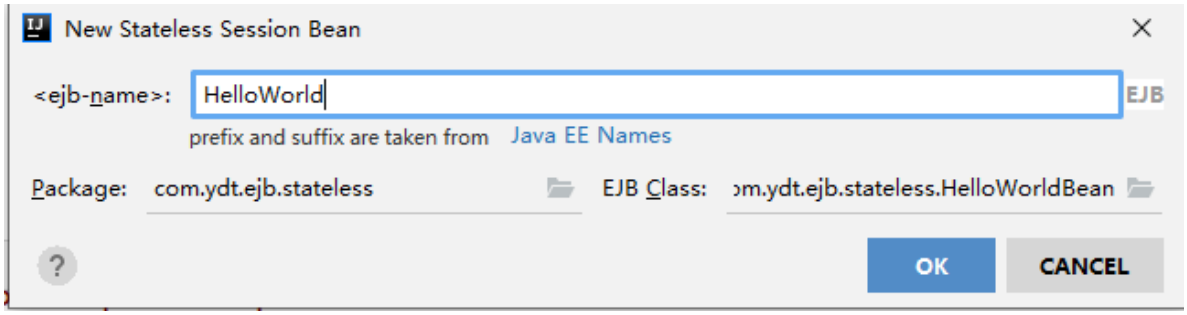


创建一个接口：HelloWorld，注意标注@Remote

```
package com.ydt.ejb.stateless;

import javax.ejb.Remote;
@Remote
public interface HelloWorld {
    public String sayHello(String name);
}
```

创建一个无状态Bean，New -> Stateless Session Bean。



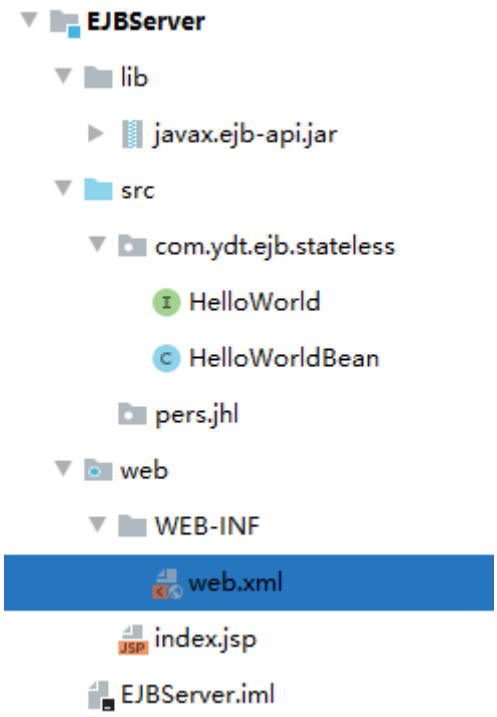
实现HelloWorld接口：

```
package com.ydt.ejb.stateless;

import javax.ejb.Stateless;

@Stateless(name="HelloWorldBean")
public class HelloWorldBean implements HelloWorld {
    public String sayHello(String name){
        return "hello," + name;
    }
}
```

项目结构如下：

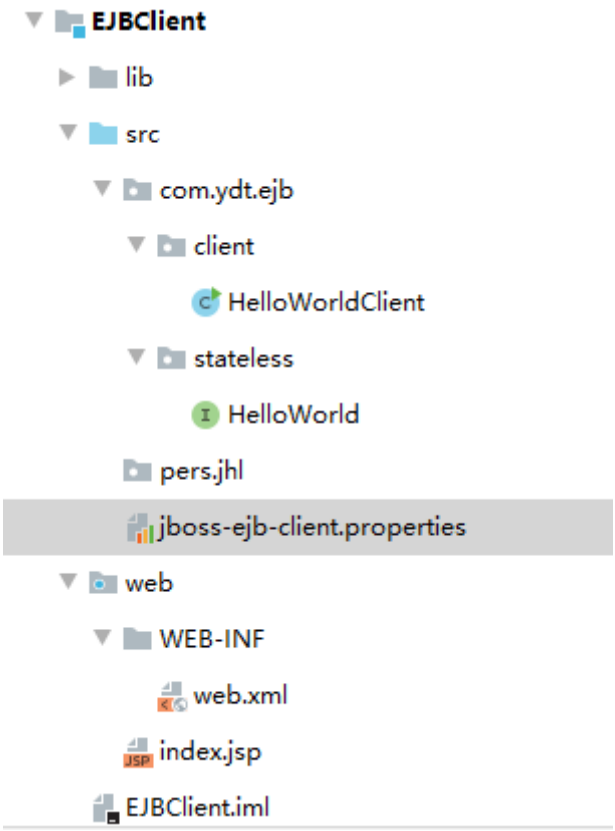


像Tomcat一样使用JBoss启动，看到如下发布的接口API即可，红色标注的为客户端调度的Bean接口：

```
java:global/EJBServer_war_exploded/HelloWorldBean!com.ydt.ejb.stateless.HelloWorld
java:app/EJBServer_war_exploded/HelloWorldBean!com.ydt.ejb.stateless.HelloWorld
java:module/HelloWorldBean!com.ydt.ejb.stateless.HelloWorld
java:jboss/exported/EJBServer_war_exploded/HelloWorldBean!com.ydt.ejb.stateless.HelloWorld
java:global/EJBServer_war_exploded/HelloWorldBean
java:app/EJBServer_war_exploded/HelloWorldBean
java:module/HelloWorldBean
```

4.4 创建EJB工程（EJBClient）

工程创建方式跟Server端一样，项目结构如下：



- HelloWorld接口跟Server端保持一致，用于接收代理Bean对象
- HelloWorldClient代码如下：

```
package com.ydt.ejb.client;

import com.ydt.ejb.stateless.HelloWorld;

import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import java.util.Properties;

public class HelloWorldClient {
    public static void main(String[] args) {
        try{
            Properties pro = new Properties();
            //设置环境变量，固定
            pro.put(Context.URL_PKG_PREFIXES,"org.jboss.ejb.client.naming");
            pro.put("jboss.naming.client.ejb.context",true);
            Context context = new InitialContext(pro);
            //这个地方就是Server端发布的Bean API接口
            HelloWorld h = (HelloWorld)
context.lookup("ejb:/EJBServer_war_exploded/HelloWorldBean!com.ydt.ejb.state
less.HelloWorld");
            System.out.println( h.sayHello("laohu"));
        }catch(NamingException e){
            e.printStackTrace();
```

```
        }

    }
}
```

- jboss-ejb-client.properties配置信息：

```
endpoint.name=client-endpoint
remote.connectionprovider.create.options.org.xnio.Options.SSL_ENABLED=false
remote.connections=default
remote.connection.default.host=localhost
remote.connection.default.port=4447 #远程服务端口
remote.connection.default.connect.options.org.xnio.Options.SASL_POLICY_NOANONYMOUS=false
remote.connection.default.username=admin #JBoss账号
remote.connection.default.password=123456 #JBoss密码
```

5、附录-IDEA搭建Spring源码环境

5.1 Gradle安装

下载：https://gradle.org/install/

Windows平台下，需要配置gradle的环境变量（Spring4.0之后源码需要使用Gradle进行部署）。

- 新增 `GRADLE_HOME` 环境变量，指向Gradle解压目录
- 配置Path环境变量:新增 `%GRADLE_HOME%\bin`
- gradle -v 查看配置是否成功：

```
C:\Users\18588>gradle -v

------------------------------------------------------------
Gradle 4.8.1
------------------------------------------------------------

Build time:   2018-06-21 07:53:06 UTC
Revision:     0abdea078047b12df42e7750ccba34d69b516a22

Groovy:       2.4.12
Ant:          Apache Ant(TM) version 1.9.11 compiled on March 23 2018
JVM:          1.8.0_131 (Oracle Corporation 25.131-b11)
OS:           Windows 10 10.0 amd64
```

5.2 下载并构建Spring源码包

下载地址：https://github.com/spring-projects/spring-framework/tree/5.2.x （下最新的）

构建源代码：

1. cmd进入到spring-framework-5.2.x项目根目录，
2. 运行gradlew :spring-oxm:compileTestJava

3. **等待构建完成，看到了BUILD SUCCESSFUL in *m *s 即可，你本地没安装git时会报异常，不管他！**

```
D:\workspaces\workspace\spring-framework-5.2.x>gradlew :spring-oxm:compileTestJava
Starting a Gradle Daemon (subsequent builds will be faster)
fatal: not a git repository (or any of the parent directories): .git

> Task :spring-oxm:genJaxb
[ant:javac] : warning: 'includeantruntime' was not set, defaulting to build.sysclasspath=last; set to false for repeatable builds

BUILD SUCCESSFUL in 1m 17s
40 actionable tasks: 26 executed, 14 from cache
Build scan background action failed.
org.gradle.process.internal.ExecException: Process 'command 'git'' finished with non-zero exit value 128
        at org.gradle.process.internal.DefaultExecHandle$ExecResultImpl.assertNormalExitValue(DefaultExecHandle.java:409)
```

5.3 更新Kolin插件

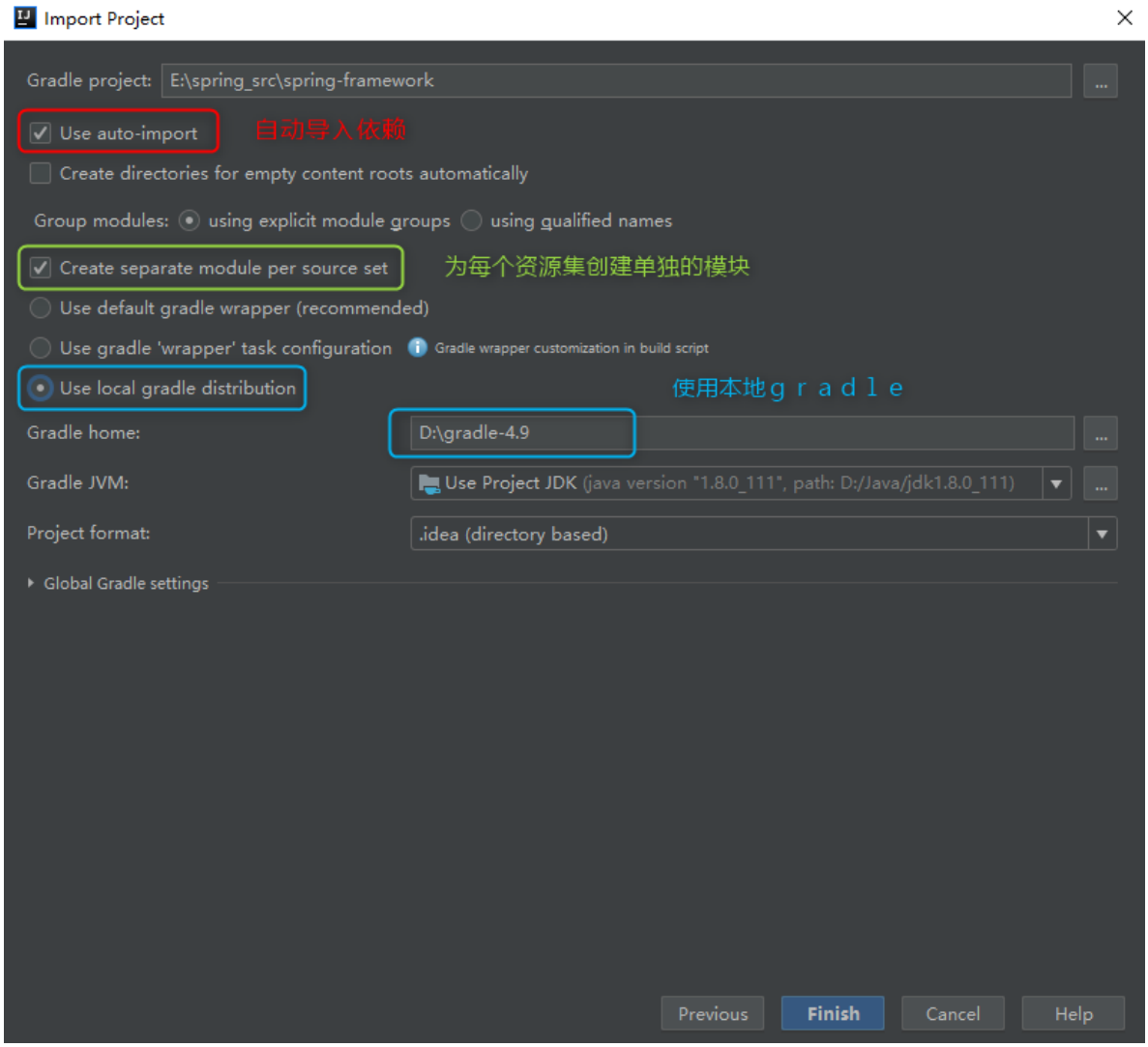
将IDEA上Kotlin插件更新到最新，File->Settings->Plugins->Kotlin,点击Update就可以了。

5.4 IDEA将源码项目导入

过程如下，并且耗时非常久（跟网络相关），更改build.gradle的maven路径为阿里云的（貌似也没太多不一样。。。。。），一直要看到所有的项目都有蓝色正方形标志才行

-----》打开IDEA，File->New->Project From Existing Sources...，选中spring-framework源码文件夹，点击OK，选择Import project from external model，选中Gradle，点击Next。

-----》点击Finish之前，可以修改一些默认的配置。



导进来的Spring源码项目本身带了很多的测试类，当然，你也可以创建Gradle工程进行测试

5.5 问题汇总

https://blog.csdn.net/gooaaee/article/details/104437902