阿里云开发者社区
ALIBABA CLOUD DEVELOPER COMMUNITY

Spring Boot 2.5 开发实战

阿里云开发者学堂推荐配套教材

作者：侠客

全面覆盖Spring Boot 2.5 新特性

实战 Redis、 MongoDB、Kafka、安全、性能监控

Spring Cloud 微服务架构的必经之路

藏经阁

阿里云开发者电子书系列

扫一扫

免费领取同步课程

钉钉扫一扫

进入官方答疑群

开发者学院【Alibaba Java 技术图谱】

更多好课免费学

阿里云开发者“藏经阁”

海量电子书免费下载

书籍简介

本书基于最新的 Spring Boot 2.5.x 版本，请更新至 2.5.x 版本。

本书全面覆盖了 Spring Boot 2.5 新特性、自动化配置原理、如何开发 REST API、实战 MySQL 数据库、Redis 高并发缓存、MongoDB 数据库、MQ 消息队列、安全机制、性能监控、高级面试题等热门知识点。

Java 高级工程师必备课程，是学习 Java Spring Cloud 微服务架构的必经之路。

目录

1. Spring Boot2.5 实战课程大纲与新特性介绍

内容简介：

一、SpringBoot2.x 实战课程大纲

二、阿里 Java 开发者学院最新课程

三、Java Spring Boot 2.5 新特性

四、Spring Boot 2.0 平台新特性

五、Spring Boot 2.4 实战 Demo

一、SpringBoot2.x 实战课程大纲

1. Spring Boot2.x 新特性

Spring Boot2.0 新特性

Spring Boot 自动配置底层原理

集成 Swagger

RESTAPI 与 MVC 网站

2. SQL 数据库

Spring Data

Hibernate 框架

Repository 模式

实战 MySQL 数据库

3. 扩展知识

MongoDB 数据库

Redis 高并发缓存

安全与性能监控

安全与性能监控

二、阿里 Java 开发者学院最新课程

1. 覆盖最新 Java 微服务架构

-Java

面向对象编程夯实基础

Java16 面向对象编程

多线程编程与锁机制

Java 垃圾回收 GC 算法

字节码机制与加载扩展

Java Web 开发框架

MySQL 数据库开发

ORM 框架实战开发

MongoDB 实战开发

-Dubbo

高并发缓存 Redis 实战

分布式架构体系

分布式 RPC 协议

Dubbo 的典型场景

淘宝双 11 服务治理

多级缓存与分布式

Dubbo 分布式架构

Dubbo3.0 优化策略

Dubbo 实战开发

云原生与容器化实战

-Spring Boot

快速开发

Spring 平台知识体系

依赖注入与 IOC 机制

Spring Boot2.5 新特性

Spring Boot 网站开发

Spring Boot API 开发

Spring Boot 性能监控

实战高并发缓存 Redis

实战开发 MongoDB

消息队列 RocketMQ

-Spring Cloud

微服务架构

微服务架构知识体系

2020 重大变化与改进

微服务注册发现机制

微服务熔断限流算法

微服务之代理网关

微服务安全身份验证

微服务之链路追踪

灰度发布与流量调度

源码解读与底层原理

-Spring Cloud Alibaba

阿里开源

阿里巴巴开源微服务

淘宝微服务架构改造

Dubbo 微服务实战

Nacos 注册发现原理

Sentinel 熔断限流

SEATA 分布式事务

分布式配置中心

负载均衡与熔断算法

异地多中心调度策略

三、Java Spring Boot 2.5 新特性

1. Java Spring Boot 2.0 框架

1）2013 年 8 月开始发布 0.50M1，2014 年 4 月发布 1.0.0 版本首先

2）SpringBoot 不是一个框架

3）之前 JavaEE 项目开发太繁琐

4）配合模板和框架来简化 Spring 项目开发

5）轻松创建具有最小或零配置的独立应用程序的方式

Spring Boot 目标主要是应用了快速开发，简化整个项目的配置和依赖工作，提升开发效率，更快速地构建应用程序。

Spring Boot 是 Spring 平台里面非常重要的基石，主要是为后续应用架构的开发设计工作做铺垫。Boot 在计算机世界中表示启动，主要目的是一站式开发。

2. Java Spring Boot 目标

Spring Boot 最初提出的设计目标里最重要的是所有东西自动化，不用成员做复杂配置，降低参数配错几率，将后续的管理、内嵌的外部服务器容器全部搞定，做到最小化依赖，最大程度降低程序运行后对人的依赖性。

四、Spring Boot 2.0 平台

Spring Boot2.0 里面有一个重大的变化叫响应式编程，相比于传统的 Serviet API 阻塞的 API，引入到非阻塞的编程模式，主要目标是提升高并发程序的吞吐量，包括底层数据库对接等。

Spring Boot 1.x 新特性

- 创建独立运行的 Spring 应用程序
- 直接嵌入 Tomcat，Jetty 或 Undertow（无需部署 WAR 文件）
- 提供运行需要的“最低”依赖项以简化构建配置
- 尽可能自动配置 Spring 和第三方库
- 提供生产就绪功能，例如指标测试，健康检查和外部 配置
- 没有代码生成，也不需 XML 配置

Spring Boot 2.x 新特性

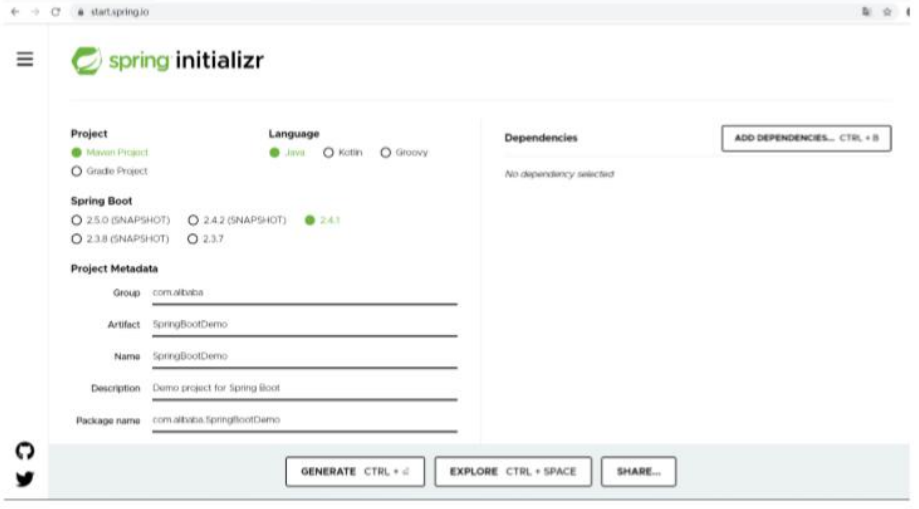
- Spring Boot
- x 不在支持 Java 7，最低 Java 8 2. Spring Boot 2.x 要求 Spring 版本 5+，Thymeleaf 3+
- Tomcat 最低版本 8.5， Jetty 9.
- 以上，Hibernate 5.2+ 4. Maven3.3+,Gradle 最低版本 3.4，提供 Gradle 插件
- 简化安全配置，默认静态资源和 Actuator 启用安全
- 增加 Reactive 响应式模块，如 Spring WebFlux

- HTTP/2 新协议支持
- Spring Boot 2.x 启用 HikariCP 替换 Tomcat 内置连接池
- 支持 Kotlin 1.2，支持性能监控 Micrometer 集成 Actuator
- 其他开发、测试、部署的小改进

五、Spring Boot 2.4 实战 Demo

开发工具可以用 idea，智能提示与模板性能方面表现优秀。

下图为网页截图 start.spring .io，这个网站主要是在线创建项目的向导，可以选择构建工具、语言、版本和各种项目的配置信息，添加必要的依赖，接着会生成一个压缩包，然后下载到本地再导入其他开发工具。



2. Spring Boot2.5 实战开发 REST API 模拟淘宝订单接口

内容简介：

一、Spring Boot 2.5 开发 快速入门

二、Spring Boot 2.5 快速开发 REST API

三、测试 Rest API 接口

一、Spring Boot 2.5 开发 快速入门

1. Spring Boot 开发环境准备

本节课讲的是 Spring Boot2.5 实战开发 REST API，模拟淘宝订单接口，从 hello world 开始，逐步把项目给复杂化。Spring Boot 属于快速开发框架，底层还是依赖于其他框架，简化了整个项目的配置，包括依赖、加载等系列过程。开发网站还依赖于底层 Servlet-API 包括 Spring MVC 的库以及 Tomcat 等相关容器组件。有的观点说 Spring Boot 是取代 Spring MVC，是错误的，并不是取代，而是更方便使用这个框架。

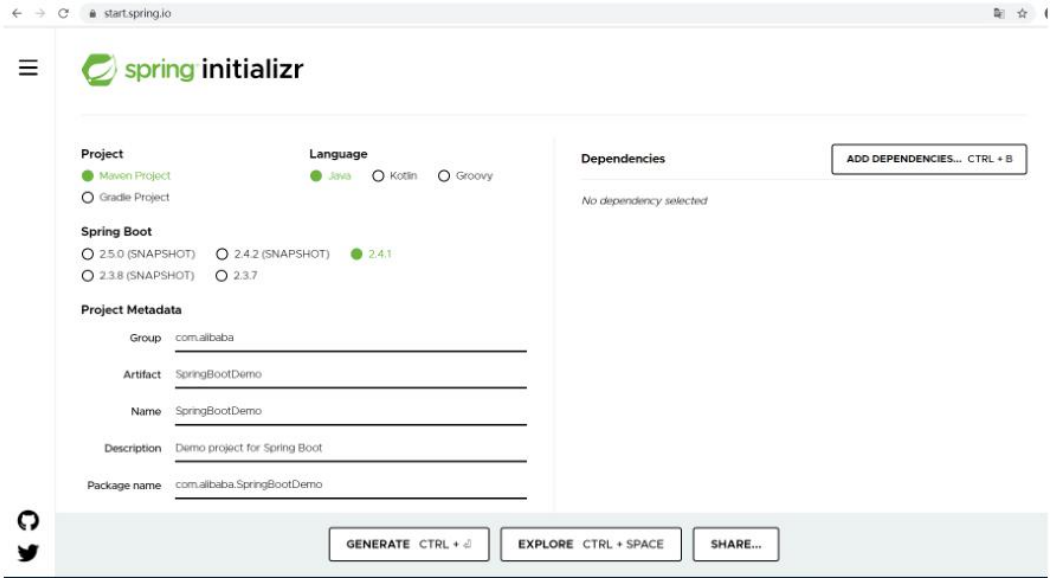
Spring Boot 开发环境准备包括：

1）Open JDK 1.8；

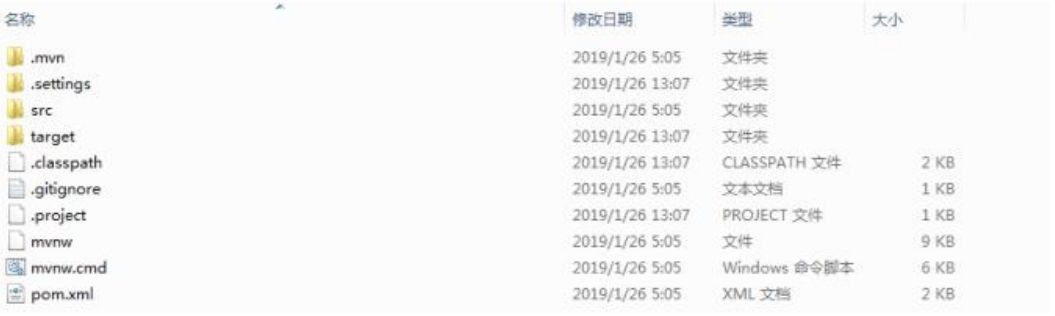
2）Eclipse 4.6+开发工具；

3）或者 IDEA 开发工具。

2. Spring Boot 2.5 实战 Demo，详细操作见第二章节



3. 下载解压缩，详细操作见第二章节



4. 简化配置，详细操作见第二章节

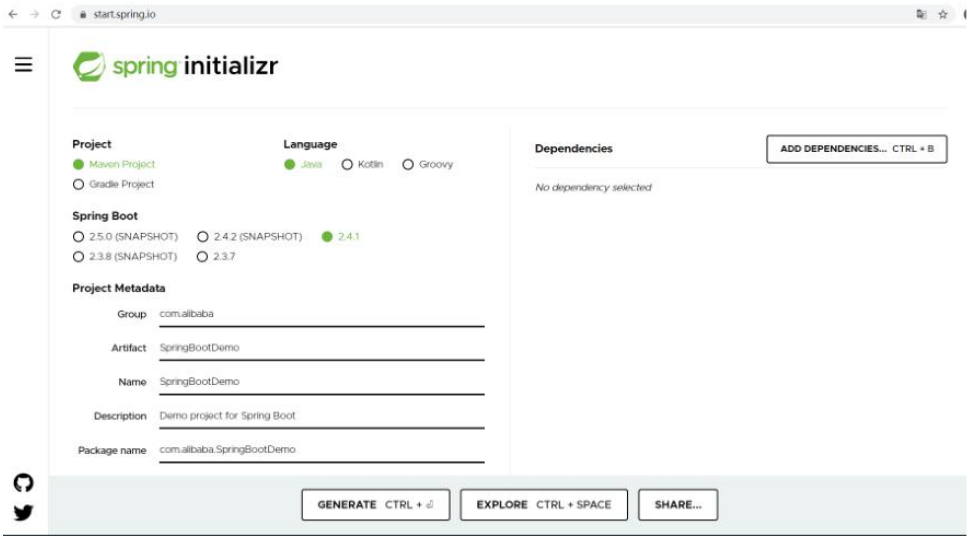
```
<properties>
    <java.version>1.8</java.version>
</properties>

<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>
```

二、Spring Boot 2.5 快速开发 REST API

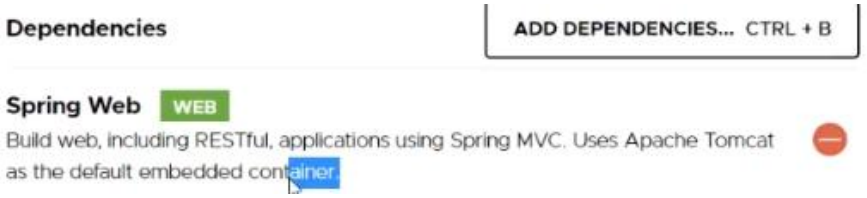
1. Spring Boot 2.5 快速开发 REST API

网页演示：

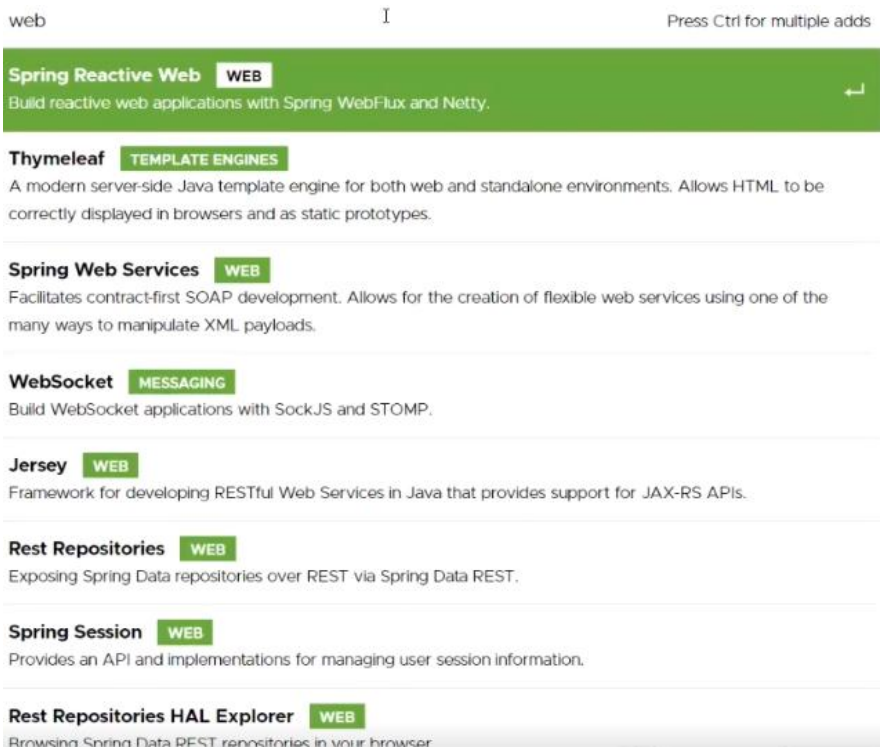
打开“start.spring.io”进入到如下面所示的界面，



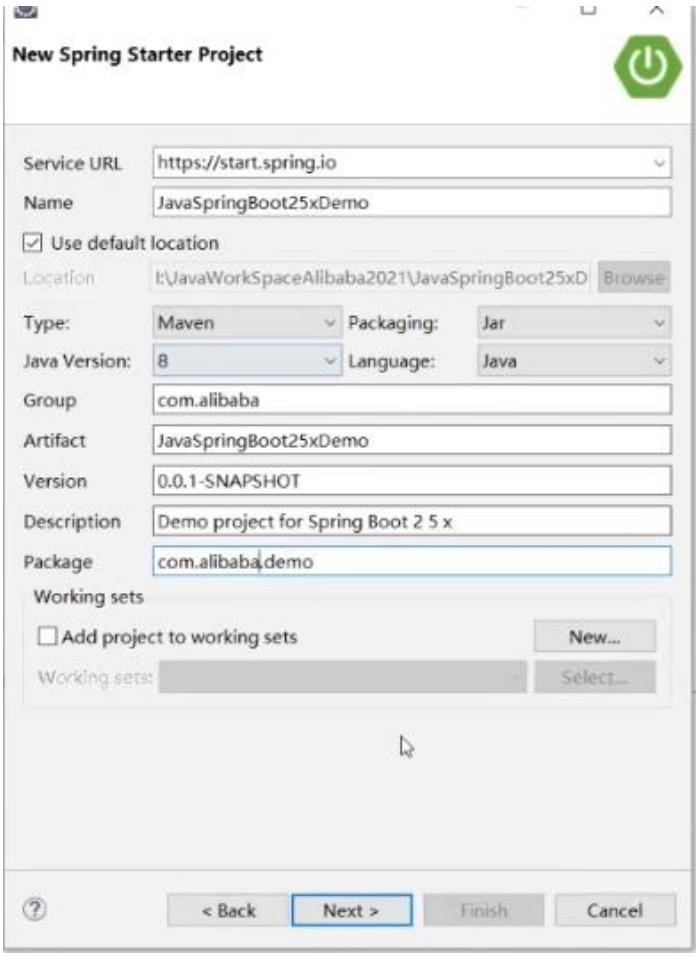
Spring Boot 可以选 2.5.0，也可以选 2.4.1；Artifact 栏输入“SpringBootDemo”；Jave 选“8”，也可以选 15，但目前大多数公司还是用“8”。现在做网站开发，一般要输入“web”关键字，底层注释使用的是“SpringMVC”,容器用的是“Tomcat” 作为默认嵌入式的 Web 容器，可以用于构建网站，也可以用于开发 RESTful API。



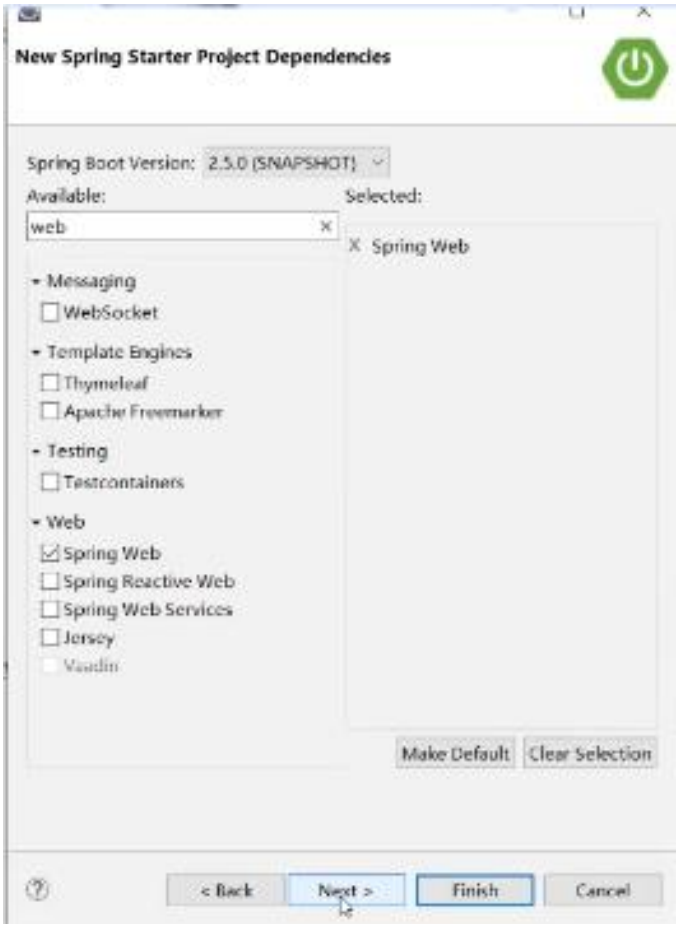
注意一下，这里面也可以选别的框架，输入 web 对应的依赖很多，与 web 相关的组件都会列出来，包括过期的组件。



这里选择传统的“SpringMVC”，点击确定生成一个压缩包，可以直接导入到直接使用，也可以解压以后再用编译器直接来做。这里讲一个比较简单的方式，直接下载安装“Spring Starter Project”插件，然后在“Other”里面选择 Spring Boot 的扩展“Spring Starter Project”然后按照项目的向导构建。



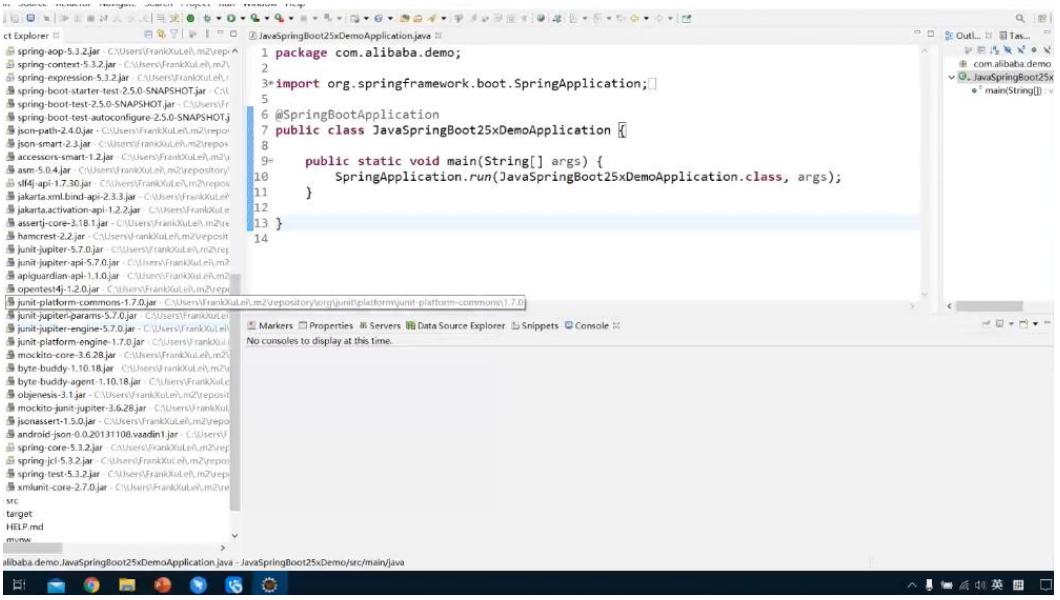
Name 输入“JavaSpringBoot25xDemo”；Java Version 版本选“8”；Group 输入“com.alibaba”;Description 输入“Demo project for Sprint Boot 2 5x”；Package 输入“com.alibaba.demo”;进入下一步，



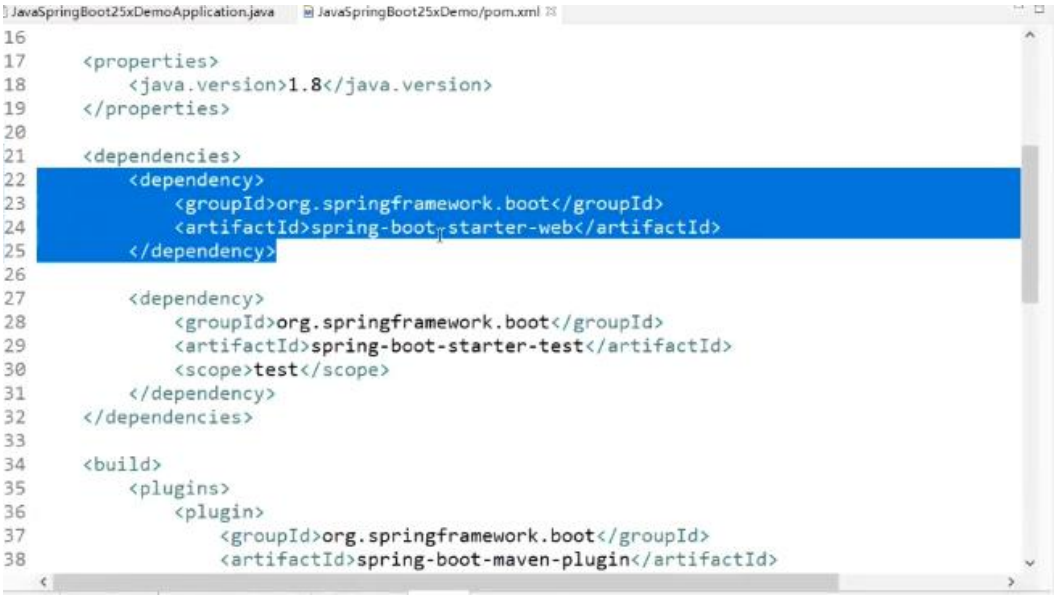
Spring Boot Versions 版本选择最新版 2.5.0 ，实际操作中建议选低一点的版本，一年以前的版本比较有保证一些，演示我们选择新的版本。Aaliable 选择“web”,web 区选“Spring Web”；下一步：

点击“Finish”,会生成一个 mvnw 项目，有 pom 文件，会有一个加载过程，第一次加载时间比较久，建议不要随意修改版本，会导致不必要的加载，下载依赖包，而且取决于网络下载的慢或者快，中间有没有中断的过程，项目启动的时候会报错等。

这时基本结构已经有了，看一下基本结构，整个项目有一个注解，用于加载配置、解析配置参数、加载必要的依赖等。打开 Maven Dependencies 可以看到很多很多的依赖，如下图所示：

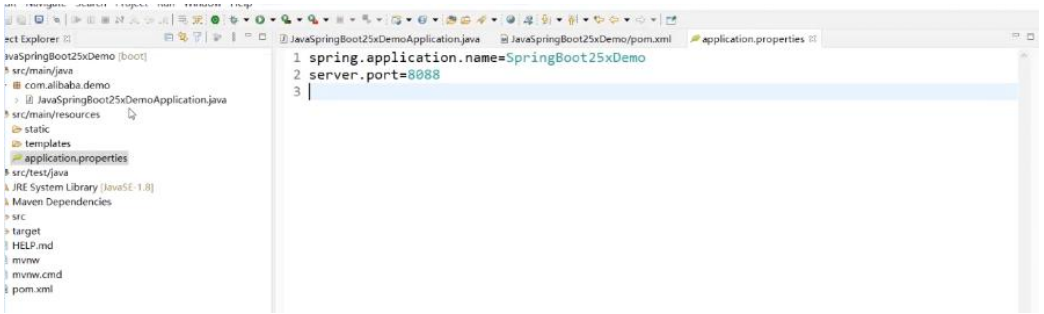


这就是傻瓜式编程很重要的原因，在 Maven Dependencies 里面配置 starter 基础依赖时，内部有依赖包清单，清单包括“spring boot starter web”等等一系列的东西，全部加载进来。表面是一个依赖，实际被解析成一列依赖列表。

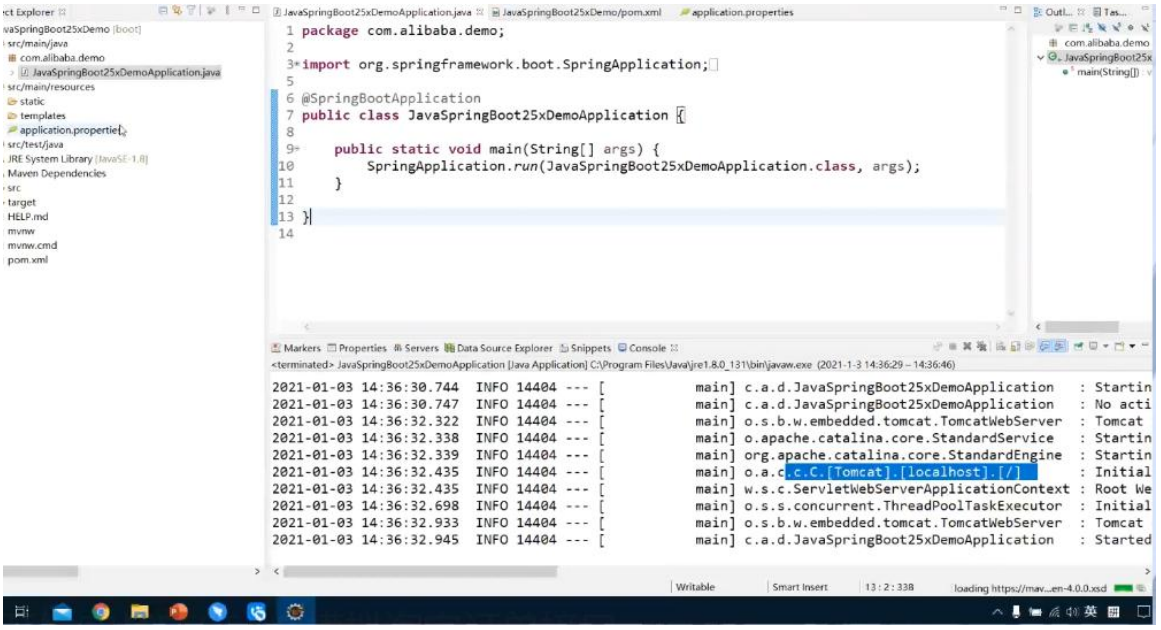


2. 修改端口

这个项目可以直接启动，嵌入的 Tomcat 属于 9.0 版本，默认端口是 8080。有可能本地已经有 Tomcat，可能会冲突，适当去改一下参数，这里有很重要的配置文件 application.properties：可以修改端口，用 server.port=8088;程序名：Spring.application.name=SpringBoot25xDemo;主机名也可以改。



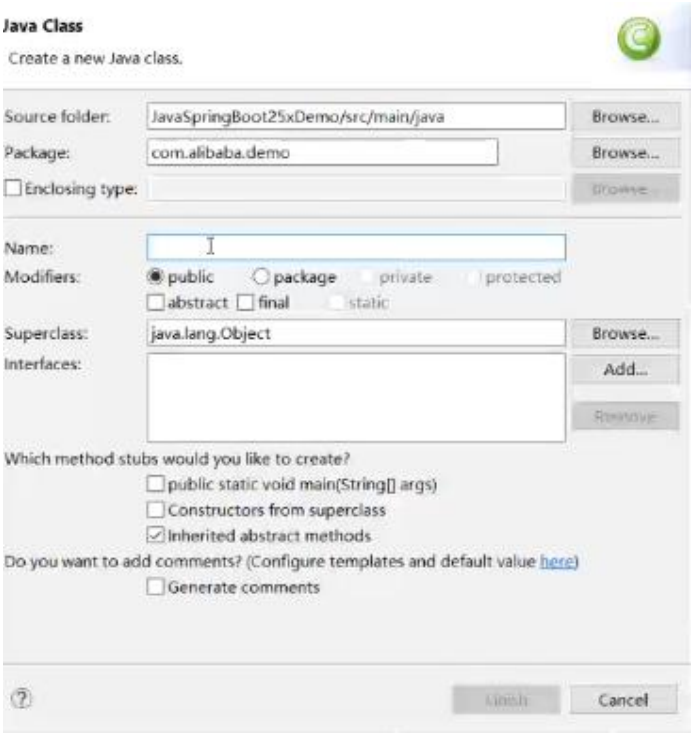
输入完成后启动，启动有几种方式，可以用解压包的形式构建，也可以右键选 Run As ,可以用 Spring Boot APP 启动，也可以选 Java Application 普通程序启动。看日志是否成功，因为程序里什么也没有，但是也可以看到“Tomcat”根地址。



三、测试 Rest API 接口

1. 浏览器测试 API

现在测试 Rest API，加 Rest 控制器，左边“com.alibaba.demo-New-Class”,打开 Jave Class ，name 输入“hello”。



加个注解，如果没有注解容易出错，输入 Hello(),造一个方法，把其暴露出去，作为 Rest API，返回字符串“Hello Spring Boot 25x”,暴露 f 地址@RequstMapping(“hello”)。

JavaSpringBoot25xDemoApplication.java | JavaSpringBoot25xDemo/pom.xml | application.properties | *HelloController.java

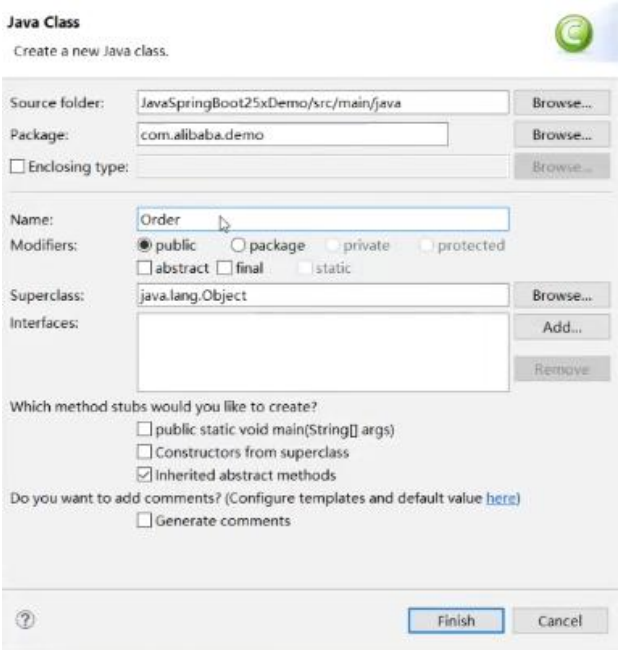
```
1  package com.alibaba.demo;
2
3  import org.springframework.web.bind.annotation.RequestMapping;
4  import org.springframework.web.bind.annotation.RestController;
5
6  @RestController
7  public class HelloController {
8
9      @RequestMapping("hello")
10     public String Hello()
11     {
12         return "Hello Spring Boot 25x";
13     }
14 }
15
```

再启动一次，右键选 Run As ,选 Java Application 普通程序启动，基本程序造构建完成了。

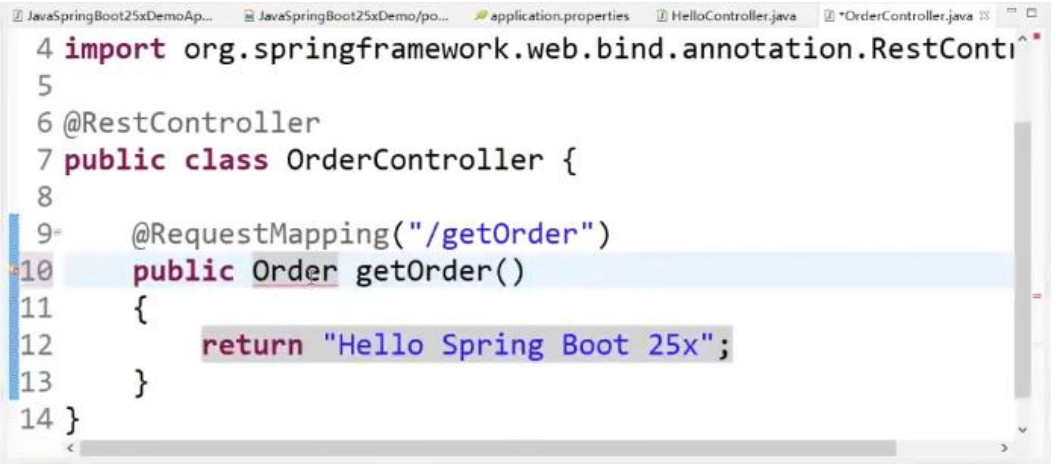
现在程序启动了没报错，打开浏览器测一下，输入“localhost:8088/hello”,返加字符串“Hello Spring Boot 25x”，这种就成功了。

2. 修改 contextpath

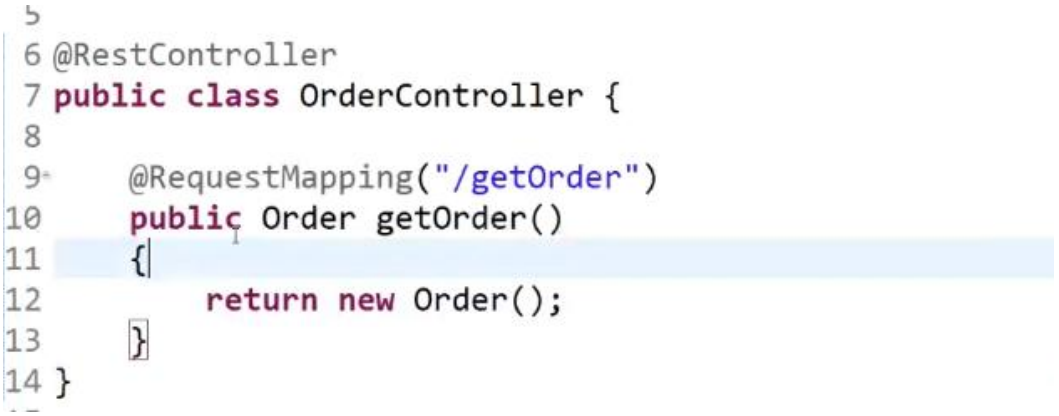
把标准项目改成复杂项目，比如加一个淘宝订单，用同样的方法，加一个 order,操作是一样的。可以把代码复制过去直接改。



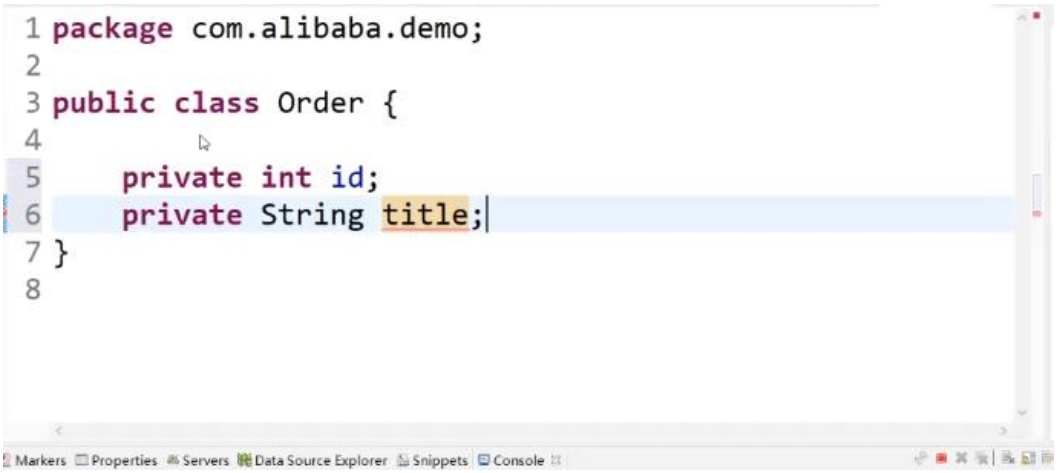
代码改成：



Order 类型没有，可以用上面的方法创建 Order 类，但是目前 Order 是空的，



在 order 里输入 private int id; private String title;订单的名字：



生成 Select getters 测试器：

把 Order 调出为使用，输入以下代码和字符串：

```
7 public class OrderController {
8
9     @RequestMapping("/getOrder")
10    public Order getOrder()
11    {
12        Order order = new Order();
13        order.setId(1);
14        order.setTitle("Iphone 18 pro Max");
15        return ;
16    }
17 }
```

再次重启一下，右键选 Run As ,选 Java Application 普通程序启动。打开网页输入“localhost:8088/getOrder”,显示输出如下字符串：

以上操作没有连真实数据库，主要用于演示，返回的是 Jackson 数据，默认用的是 Jackson 序列化，阿里开源的叫 Fastjson 序列化。本节课讲的是 Spring Boot2.5 实战开发 REST API，模拟淘宝订单接口，从 hello world 开始，然后改成 get older，逐步把项目给复杂化。

3. Spring Boot2.5 自动化配置 Autoconfig 底层原理

内容简介:

一、Java Spring Boot 2.5 自动化配置机制解析

二、Spring Boot 2.0 自动化 配置机

三、Spring Boot 2.0 自动化 配置机

四、Spring Boot 2.5 自动化 配置机制流程

一、Java Spring Boot 2.5 自动化配置机制解析

1. Java Spring Boot 目标

Spring Boot 的设计的目标，就是叫敏捷式开发，简化整个 Java 应用程序的开发过程，首先是自动化的依赖的管理工作，配置相对简单，依赖解析也是自动化的，容器，包括中心点配置尽量完成，尽量通过少代码而实现整个 Java 项目的管理配置工作，这是它的一个初衷，对于提升开发效率简化配置过程，非常有帮助。

2. Bean 配置

因为底层封装的时候非常隐蔽，不容易知道其中的一些实验原理，实际为了做一个 Java 的网站，需要改善了代码，大量/bean 的配置，文件得修改，十分复杂的，但是现在 Spring Boot2.0，这些工作已经都不需要去这么来做了，越来越简单。

二、Spring Boot 2.0 自动化 配置机制

1. Spring Boot 2.0 自动化配置机制

底层得具体操作过程，自动化配置就是整个审核的项目会自动完成一个配置的解析、包的加载过程、下载加载过程以及注入过程，甚至包括容器环境的一些配置，最核心的是在 2.0 以后有个重点 SpringBootApplication ，背后是其他几个注解的一个捆绑体，另外一个观点是提出了叫非侵入式，但是完全做非侵入式还是比较难，但是有两种选择，一通过配置文件参数配置的方式来完成，二通过代码的方式来做。

以下是在 SpringBoot2.0 以后做的一些自动化配置的机制。

1）Spring Boot auto configuration
2）Spring Boot 自动配置
3）尝试根据 classpath 的 jar 依赖自动配置 Spring 应用。
4）@AutoConfiguration 注解 过期
5）@Configuration 注解
6）@EnableAutoConfiguration 注解
7）@SpringBootApplication 注解 2.0 版本新增
8）Auto-configuration is non-invasive 非侵入式
9）也可以禁用自动配

2. @SpringBootApplication 注解等于三大注解

上节课所写的程序里面，实际底层背后与配置相关的有三大注解：

- **@EnableAutoConfiguration:**enable Spring Boot’s autoconfiguration mechanism
- **@ComponentScan:** enable @Component scan on the package where the application is located (see the best practices)
- **@Configuration:** allow to register extra beans in the context or import additional configuration classes

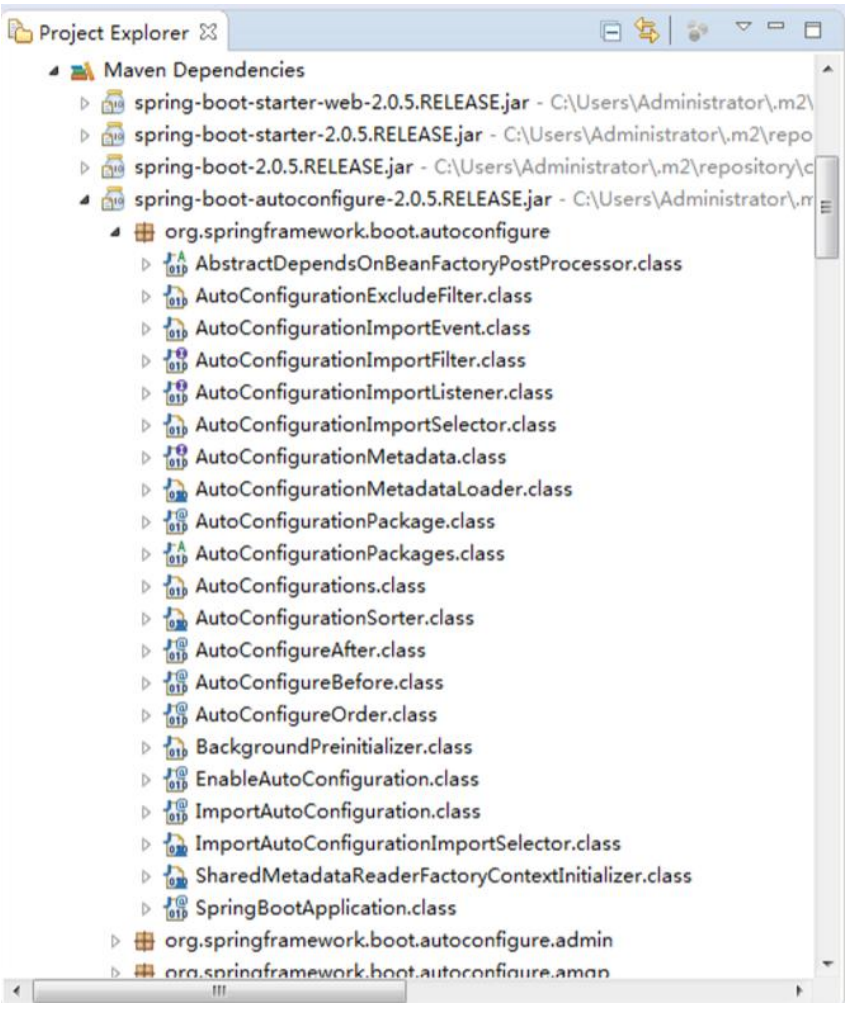
configuration 注解，在早期的使用 MVC 里面，Spring 框架里面已经拥有,用来实现自动化配置的加载的过程，可以去设置一些通过类型、配置文件来做。

ComponentScan 可以去做一些组件的扫描，如安全的组件，数据访问层的业务流程的组件，甚至其他一些模型的实体的组件，都可以标记完以后由它来进行扫描，如指定一个包，去特定包下面进行扫描那些组件。

EnableAutoConfiguration 在 SpringBoot 里就提供了，主要是告诉当前的程序启用什么，启用自动化配置的一个机制，现在 SpringBootApplication 注解可以直接来实现，大大简化了数据配置过程。

3. 自动化配置机制核心

- spring-boot-autoconfigure.jar
- spring.factories



之前讲过的项目，在 SpringBoot 依赖包里面有 autoconfigure 依赖包，它里面实际对整个 SpringBoot 的加载做了很多扩展，在配置、解析、加载等等一系列过程中的话，埋下点，可以再进行工作的扩展，如在配置刚加载的时候，拦截校验，配置下载完成以后，可以去修改它配置参数等等一系列工作。

4. @SpringBootApplication 注解

```
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Inherited
@SpringBootConfiguration
@EnableAutoConfiguration
@ComponentScan(excludeFilters = {
        @Filter(type = FilterType.CUSTOM, classes = TypeExcludeFilter.class),
        @Filter(type = FilterType.CUSTOM, classes = AutoConfigurationExcludeFilter.class) })
public @interface SpringBootApplication {

    /**
     * Exclude specific auto-configuration classes such that they will never be applied.
     * @return the classes to exclude
     */
    @AliasFor(annotation = EnableAutoConfiguration.class)
    Class<?>[] exclude() default {};
```

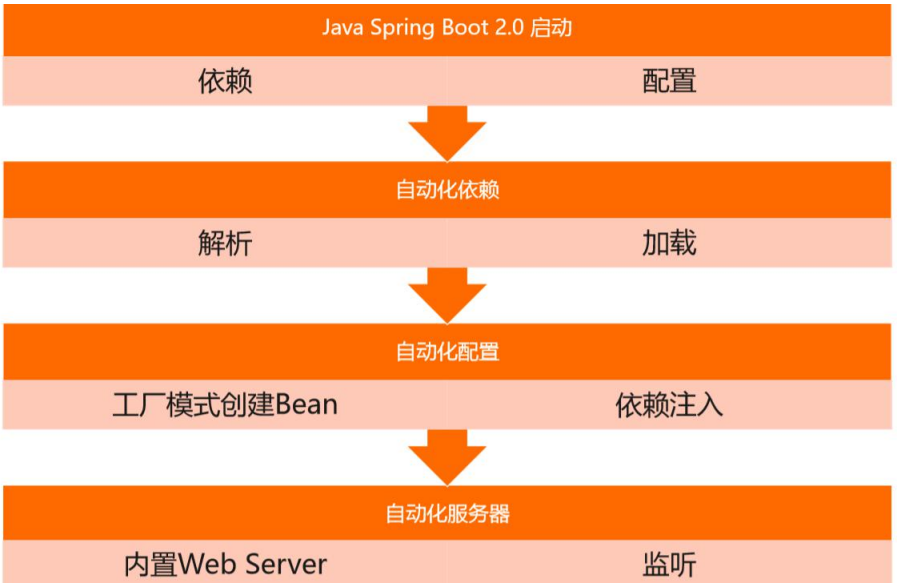
5. @SpringBootApplication 注解机制

- @SpringBootApplication 注解= @SpringBootConfiguration+ @EnableAutoConfiguration + @ComponentScan 之和。
- @Configuration 将该注解类标记为应用程序上下文的 bean 来源。
- @EnableAutoConfiguration 告诉 Spring Boot 自动配置添加 bean。
- 通常手动会为 Spring MVC 应用程序添加@EnableWebMvc。
- 但 Spring Boot 会在类路径上看到 spring-webmvc 时自动添加它该注解。
- 为 Web 应用添加并启用关键特性，例如设置 DispatcherServlet。
- @ComponentScan 告诉 Spring 扫描组件，配置和服务，控制器。

三、Spring Boot 2.5 自动化 配置机制流程

1. Java Spring Boot 2.x 启动过程

Spring Boot 成为启动过程中，配置这个选项是非常重要，当然也有一些默认参数，显示配置和默认配置，当它内置的有一个默认的配置清单，加一个 starter 的一个依赖，是背后有一个默认清单，然后逐步去把所有的默认的配置，版本号对应版本的包给拉进来。



2. 面试题：自动化装配

@SpringBootApplication=

@EnableAutoConfiguration +

@Configuration

@ComponentScan

本质上还是早期 Spring 实现的叫配置的一个注解 configuration，加上组件扫描等，启用制造位置，只告诉当前的程序要启用一系列默认的默认的策略，有清单要找一下，

有可能有拦截的代码，想办法给去检查一下是否执行相关逻辑的扩展代码。

3. AutoConfigurationPackages

1）@EnableAutoConfiguration 在里面担任的角色是非常重要的，实际是另外一个扩展点，针对 SpringBoot 的一个扩展点。

2）AutoConfigurationPackages.Registrar 注册存储客 户端配置包列表的 bean。

3）便于以后使用。

4）pring 引导在内部使用此列表，例如 在 springboot-autoconfigure 的数据访问配置类中。

5）可以通过静态方法 AutoConfigurationPackages.get （BeanFactory）访问此列表：

包加载有一个专门的 AutoConfigurationPackages.Registrar 管理的一个类型，属于 SpringBoot 的一个扩展。

4. ImportSelector

1）导入选择器

2）@Import （EnableAutoConfigurationImportSelector.class）

3）此批注负责引导自动配置机制

4）EnableAutoConfigurationImportSelector 实现 DeferredImportSelector。

5）个选择器实现使用 Spring 核心 SpringFactoriesLoader.loadFactoryNames（）

6）它从 META-INF / spring.factories 加载配置类

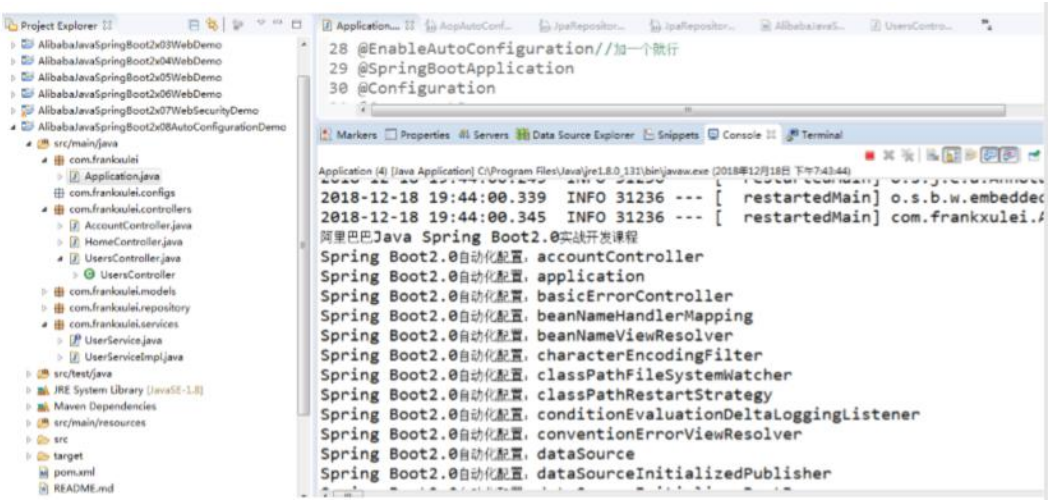
四、Spring Boot 2.0 监控自 动化配置 Bean

1. 监控自动化配置 Bean 代码

监控一下整个 Bean 的注入过程，如下图所示，

```
@SpringBootApplication
@Configuration
@ComponentScan
public class Application implements CommandLineRunner {

    public static void main(String[] args) {
        ApplicationContext ctx = SpringApplication.run(Application.class, args);
        //日志打印系统自动化配置的bean
        String[] beanNames = ctx.getBeanDefinitionNames();
        Arrays.sort(beanNames);
        for (String beanName : beanNames) {
            System.out.println("Spring Boot2.0自动化配置: "+beanName);
        }
    }

    @Override
    public void run(String... arg0) throws Exception {
        // TODO Auto-generated method stub
        System.out.println("阿里巴巴Java Spring Boot2.0实战开发课程");
    }
```

举例：



有些是根本用不到的，只需要请求路由，请求的处理，然后加上一系列的虚拟化反应的话就可以，看到上面那个机制，它加载了可能上百个 Bean，但是实际是没有必要的。

使用波段的本身，自动化的配置的过程，确实是大大减化了配置工作，提升了开发的效率。

4. Spring Boot2.5 使用 Spring Data 链接 MySQL 数据库

内容简介：

一、Spring Data 简化 MySQL 数据访问

二、Spring Boot 2.5 实战 MySQL 数据库

三、Demo

一、Spring Data 简化 MySQL 数据访问

1. Spring Data 新特性

Spring 在 Spring boot 之后再应用开发、微服务架构以及数据链接上都提供了专门的框架，大大简化开发工作，提升开发的效率。

Spring Data for MySQL 有很多技术可以用，比如 JDBC、JDBC Template、RM 框架或者 Hibernate My Business。

Spring Data 会整合框架，简化整个框架的配置。这里面有个非常重要的 Spring Data 的子集叫 JPA，实际就是加上了一个持久化的 API，它其中有一块针对 MySQL 封装底层的 Hibernate，也可以切换成 My Business。

Spring Data 新特性

1）快速数据访问框架，提供统一的编程模型

2）强大的 repository 仓储和自定义对象映射 ORM 抽象

3）从 repository 方法名称派生动态查询接口

4）实现 Domain 域基类提供基本属性

5）支持透明审计日志（创建，最后更改）

6）可以自定义 repository 代码

7）通过 JavaConfig 和自定义 XML 命名空间轻松实现 Spring 集成

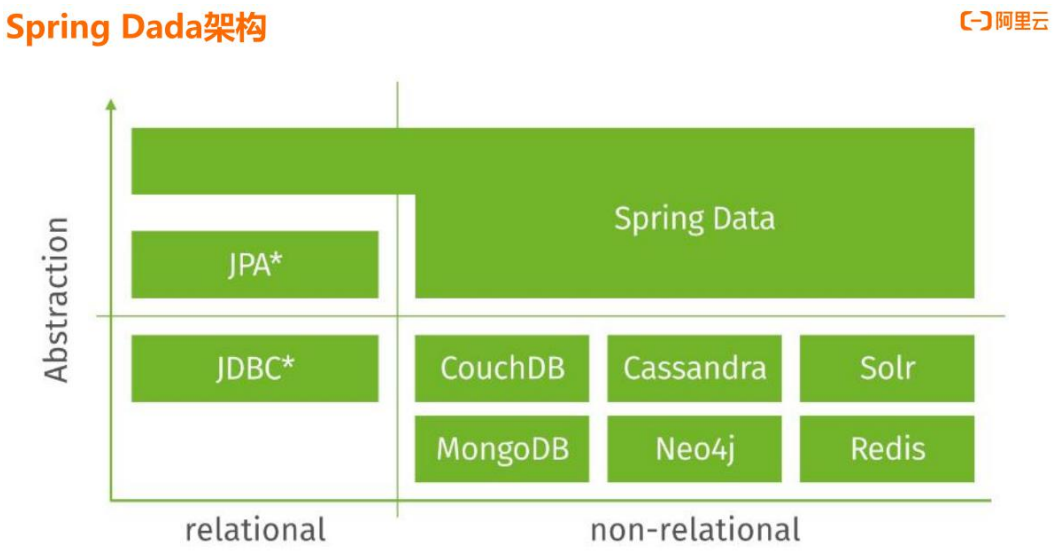
8）与 Spring MVC 控制器的高级集成

9）跨库持久性的实验支持

Spring Data 针对各个数据源提供了统一的编程模型，其中有一个设计模式叫仓储模式，仓储模式在数据访问层又做了一层封装，主要针对各种不同的数据库提供统一的操作，有些默认接口直接生成不用再进行配置了。这个操作也可以结合其他的分层模型来进行整合。

2. Spring Data 架构

Spring Data 主要是做各种不同的数据源的对接工作。有传统的关系型数据库也有非关系型数据库等等都可以和它进行集成。



Spring Data 架构主要分成了关系型数据库和关系型数据库。JPA 底层使用 JDBC。

本次主要对关系型据库的一系列操作，后面对于访问 Redis 以及 MongoDB 后续课程也有对应的实战案例。

3. Spring Data 核心模块

1）Spring Data Commons - 支持每个 Spring Data 模块的 Core Spring 概念。

2）Spring Data JDBC - 对 JDBC 的 Spring Data 存储库支持。

3）Spring Data JDBC Ext - 支持标准 JDBC 的数据库特定扩展，包括对 Oracle RAC 快速连接故障转移的支持，AQ JMS 支持以及对使用高级数据类型的支持。

4）Spring Data JPA - JPA 的 Spring Data 存储库支持。

5）Spring Data KeyValue - 基于映射的存储库和 SPI，可轻松构建用于键值存储的 Spring Data 模块。

6）Spring Data LDAP - 对 Spring LDAP 的 Spring Data 存储库支持。

7）Spring Data MongoDB - 基于 Spring 的对象文档支持和 MongoDB 的存储库。

8）Spring Data Redis - 从 Spring 应用程序轻松配置和访问 Redis。

9）Spring Data REST - 将 Spring Data 存储库导出为超媒体驱动的 RESTful 资源。

10）Spring Data Apache Cassandra - 轻松配置和访问 Apache Cassandra 或大规模，高可用性。

11）Spring Data Apache Geode - 轻松配置和访问 Apache Geode。

12）Spring Data Apache Solr - 为面向搜索的 Spring 应用程序轻松配置和访问 Apache Solr。

13）Spring Data Pivotal GemFire - 轻松配置和访问 Pivotal GemFire。

4. Spring Boot2.5 实战 MySQL

1）Spring JDBC and JdbcTemplate

2）Spring Data JPA and Hibernate framework

3）Spring Data 简化连接不同的数据库

4）使用 Spring Data JPA 框架连接 MySQL

5）当然也可以使用原始的 JDBC

6）默认底层使用 Hibernate 框架

7）支持 Repository 仓储模式

8）引入最重要的 2 个包

9）spring-boot-starter-data-jpa

10)mysql-connector-java

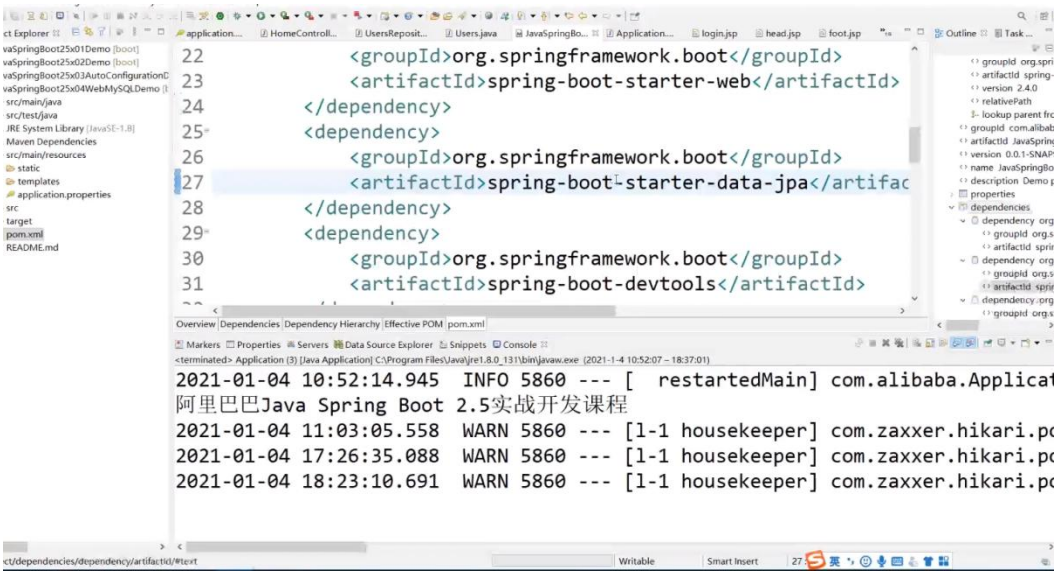
JPA 默认一个 RM 框架叫 Hibernate 框架。我们在对接的时候，需要大家去引几个包。作为 Spring boot 严格来说是有个傻瓜式编程，要简化配置。但是我们手动引入一个叫 JDBC 的包，然后再去配置数据库链接参数，数据库地址以及用户密码这些参数还是要自己设置。

对于初始化依赖，可以使用 starter-data-jpa 这样的一个依赖包。这里面还涉及到 JdbcTemplate，这是 Spring 提供了一个非常重要的数据库操作的接口。

5. Spring Data JPA 框架

- Spring Data JPA 简化数据访问层的开发工作
- 基于 Spring 和 JPA 构建存储库的完美支持
- 支持 Querydsl 谓词，从而支持类型安全的 JPA 查询
- Domain 类的透明审核
- 分页支持，动态查询执行，集成自定义数据访问代码的能力
- 在引导时验证@Query 带注释的查询
- 支持基于 XML 的实体映射
- 引入@EnableJpaRepositories，基于 JavaConfig 的存储库配置

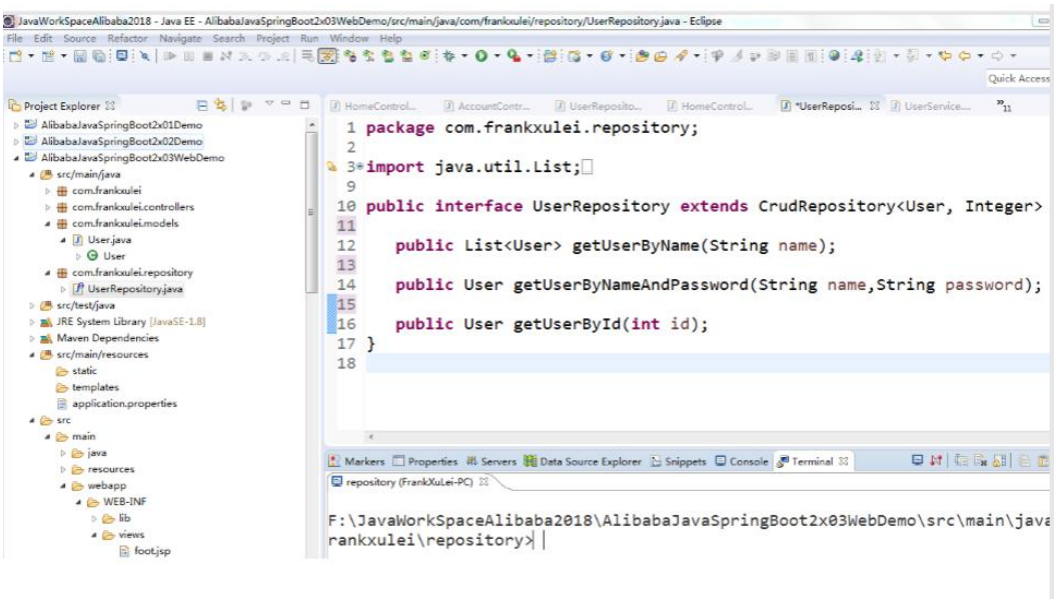
在定义接口的时候，特殊的查询可能需要编写一些特殊的语句，下面来看一下具体的一个实现。



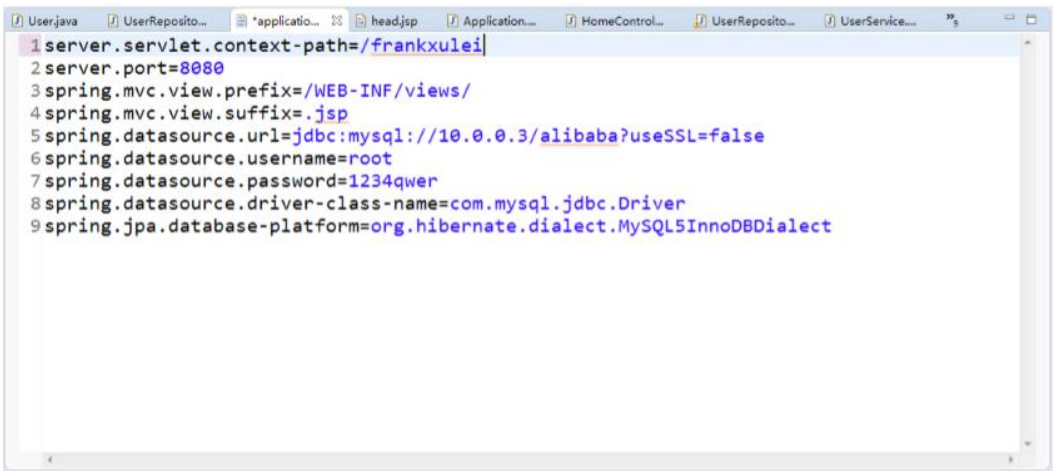
加入数据库访问链接，这里面有个 spring data starter 的 JPA。加进来以后我们从数据库底层手动引一下 MySQL-connector，里面 devtools 主要做自动化调试，方便进行程序的开发配置。改完代码以后不用重启，就会自动重新加载变化的数据配置文件。

Database platform 在这里面指的是 hibernate。Model 在这里面指的是实体，和数据库直接对应。相比之前 Spring data 对应的数据源要配置一下。代码中有几个重要的参数分别是 url、username 以及 password。

Spring Data Jpa 是链接 MySQL 数据库重要的组件，在配置的时候要注意底层的 MySQL connection。



配置 MySQL 参数:



POM 配置:

```xml
        <relativePath /> <!-- alibaba java spring boot实战课程 -->
    </parent>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
        <java.version>1.8</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
        </dependency>
```

仓储 Repository 泛型接口:

```java
public interface CrudRepository<T, ID> extends Repository<T, ID> {

    /**
     * Saves a given entity. Use the returned instance for further operations as th
     */
    <S extends T> S save(S entity);

    /**
     * Saves all given entities.
     */
    <S extends T> Iterable<S> saveAll(Iterable<S> entities);

    /**
     * Retrieves an entity by its id.
     */
    Optional<T> findById(ID id);

    /**
     * Returns whether an entity with the given id exists.
     */
    boolean existsById(ID id);

    /**
     * Returns all instances of the type.
     */
    Iterable<T> findAll();

    /**
     * Returns all instances of the type with the given IDs.
     */
    Iterable<T> findAllById(Iterable<ID> ids);

    /**
     * Returns the number of entities available.
     */
    long count();

    /**
     * Deletes the entity with the given id.
```

第 4 课：

Java 高级面试题

1. Spring Data JPA CRUD 增删改查 REST API
2. 如何基于 Spring Boot+Spring Data 实现登录注册？
3. Spring Data 为什么只使用接口声明就可以访问数据库？
4. 默认 Spring Data JPA 使用 Hibernate，如何配置连接池？

5. Spring Boot 2.5 实战 MongoDB 数据库与面试题

内容简介：

一、Spring Boot 2.5 实战 MongoDB 数据库

二、NoSQL 排名第一 MongoDB

三、安装 MongoDB 数据库

四、Spring Data 实战 MongoDB 数据库

五、本节课高级面试题

一、Spring Boot 2.5 实战 MongoDB 数据库

阿里云大学 MongoDB 的高级实战课程，涵盖入门到各种高级主题、索引、存储引擎及各种对接包括日志加密、安全身份验证、高可用集群，分片集群等。

《阿里巴巴 MongoDB4.0 高级实战》

1）MongoDB 数据库入门：MongoDB 概览、4.0 新特性、下载安装、Shell 连接及基本操作等。

2）MongoDB 数据库数据查询与分析：MongoDB 查询命令、分析、聚合等。

3）MongoDB 数据库核心知识：MongoDB 数据库操作、集合、存储引擎、数据模型等。

4）MongoDB 数据库管理：MongoDB 数据相关操作、数据备份与恢复等。

5）MongoDB 数据库性能分析与调优：MongoDB 索引、算法、查询计划等。

6）MongoDB 与 Java 开发实战：基于 Java Spring Boot 和云数据库 MongoDB 开发 HTML5 博客应用。

7）MongoDB 数据库排错日志分析：MongoDB 日志收集、经典问题解析、常用问题排查工具等。

8）MongoDB 数据库安全机制：MongoDB 身份验证、加密、典型机制等。

9）MongoDB 数据库 HA 高可用集群架构：主从复制、读写分离、自动化故障转移、HA 集群，阿里云数据库集 群等 。

10）MongoDB 数据库运维：数据库维护、升级，云数据库 MongoDB 版运维、监控工具，数据库容灾方案等。

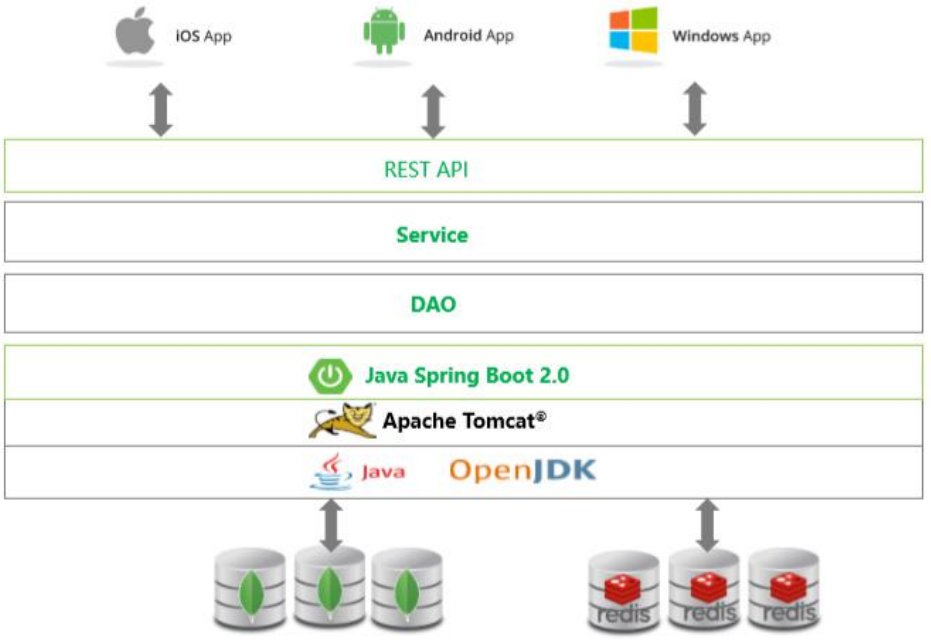
11）MongoDB 优化实战案例：讲解 MongoDB 的索引原理，以及常见的优化手段，并分析一些具体的优化案例。

12）MongoDB Sharding 集群原理与架构优化：讲解 Sharding 集群原理、架构设计方法，以及常见的架构优化手段。

13）官方网站：https://edu.aliyun.com/workshop/3/course/1044

二、NoSQL 排名第一 MongoDB

1. 移动互联网架构



2. MongoDB 简介

MongoDB 是文档型数据库，较灵活，容易做集群搭建，在互联网公司运用广泛。

1）NoSQL 排名第一，BAT 互联网公司必备

2）分布式数据库

3）由 C++语言编写，特点是高性能、易部署、易使用、存储数据非常方便

4）旨在为 Web 应用提供可扩展的高性能数据存储解决方案

5）MongoDB 由 10gen 团队所开发，于 2009 年 2 月首度推出

6）MongoDB 开源、跨平台

7）支持 Windows、Linux、OS X 和 Solaris 系统

8）MongoDB 最新版本为 4.0，支持跨文档事务

3. MongoDB 优点

1）灵活的数据模型

2）便于横向拓展

3）自动分片存储

4）支持分布式查询

5）集成内存缓存

6）高性能高并发

4. MongoDB 的典型行业案例

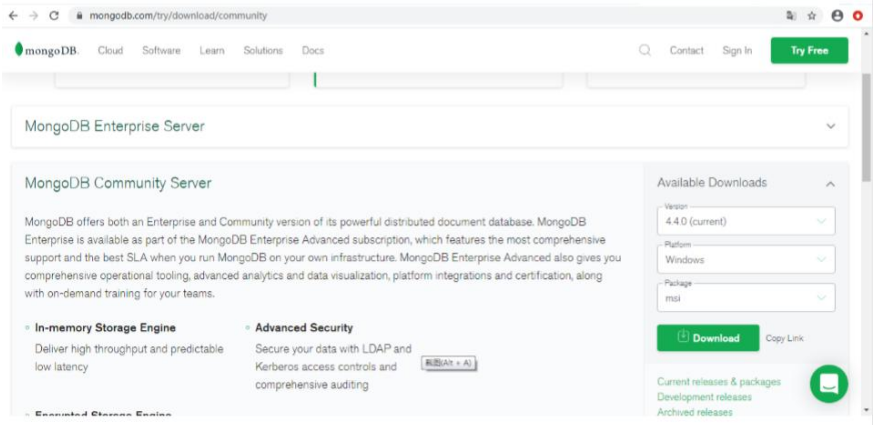
5. MongoDB 版本特性

版本	关键特性	建议
2.X	index、writeConcern、readPreference	强烈建议升级
3.0	Pluggable Storage Engine、Wiredtiger、improved mmapv1	建议升级
3.2	Raft 协议、文档校验、部分索引、inMemory、$lookup	建议升级
3.4	并行复制、sharding迁移改进、collation、$facet、$graghLookup	强烈建议使用
3.6	安全、并行性能、$lookup、Online 维护（在线oplog维护、在线添加认证）	已经发布
4.0	分布式事务Transaction	已经发布

推荐大家用 3.0 以后的版本,因为 3.0 以后默认 Wiredtiger，性能、稳定性更强大，4.0 以后基本上也支持分布式事务 Transaction，包括分片集群、复制集群事务，其他并行异步复制等，提升了针对大容量数据库迁移的优化。

二、安装 MongoDB 数据库

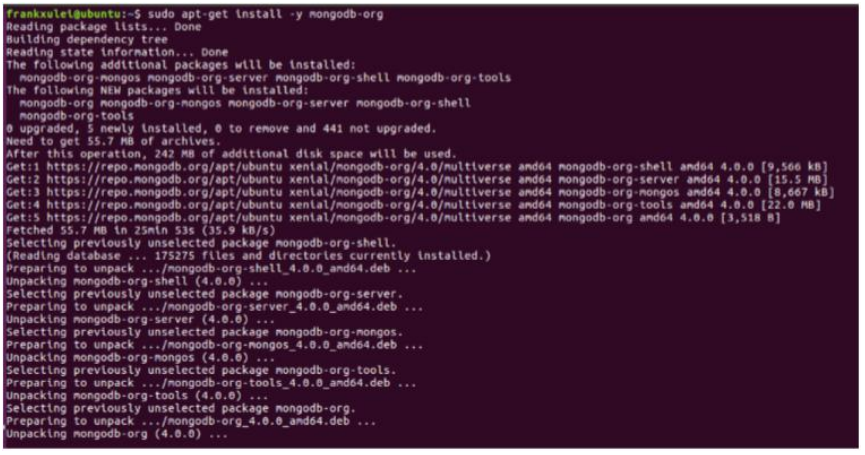
1. 下载 MongoDB4.4



官方下载，windows 可以直接下，Linux 直接用命令，安装后启动 MongoDB 服务。

2. 安装最新 MongoDB

3. Linux 安装 MongoDB4.0



4. 可视化管理工具

- Robomongo
- Robo 3T
- Compass

5. MongoDB 操作命令

- > show dbs
- admin 0.000GB
- config 0.000GB • local 0.000GB
- > show users
- > show collections
- > use Alibaba
- switched to db alibaba
- > db.users.insert({"name":"Java"})
- WriteResult({ "nInserted" : 1 })
- > db.users.insert({"name":"mongo","age":18})
- WriteResult({ "nInserted" : 1 })
- > db.users.find() • { "_id" :ObjectId("604ca8c13bfab2e14d4927d4"), "name" : "Java" }
- { "_id" : ObjectId("604ca8e33bfab2e14d4927d5"), "name" : "mongo", "age" : 18 }
- > db.users.insert({"name":"frank","password":"1234","age":18})

- WriteResult({ "nInserted" : 1 }) • > db.users.find()
- { "_id" : ObjectId("604ca8c13bfab2e14d4927d4"), "name" : "Java" }
- { "_id" : ObjectId("604ca8e33bfab2e14d4927d5"), "name" : "mongo", "age" : 18 }
- { "_id" : ObjectId("604ca9623bfab2e14d4927d6"), "name" : "frank", "password" : "1234", "age" : 18 }

四、Spring Data 实战 MongoDB 数据库

1. Spring Data 2.5 MongoDB 新特性

1）简化 Java 的 MongoDB 数据库开发 API

2）提供一致的基于 Spring 的编程模型

3）Spring configuration 支持@Configuration classes or an XML namespace

4）方便编写 Repository 仓储模式的 DAO 层代码

5）自动实现 Repository interface 的 CRUD 常用操作

6）自动进行 POJO 和 MongoDB 文档数据的映射转换， Spring’s Conversion Service

7）可以自定义扩展方法

8）MongoTemplate helper class 提升 MongoDB 开发效率

9）Low-level mapping using MongoReader/MongoWriter abstractions

10）Java based Query, Criteria, and Update DSLs

11）Log4j log appender

12）GeoSpatial 集成

13）Map-Reduce 集成

14）JMX administration and monitoring

15）CDI support for repositories

16）GridFS 支持

2. Repository 仓储层代码

```
public interface BlogRepository extends MongoRepository <Blog,
objectId> {
   public Blog findById(objectId id);
   public void delete (objectId id);
   public List<Blog> findAll();
 }
```

Repository 仓储层代码,实际大大的简化了整个数据库编程，但是底层基础还是需要大家去使用，作为应用开发人员，使用接口越方便越简单效率越高，但是底层实现需要我们花时间去研究学习。

五、本节课高级面试题

高级面试题

1. Spring Data for MongoDB 映射机制
2. 官方驱动是什么

3. MongoDB 存储什么数据？NoSQL,面向文档 JSON
4. MongoDB 优缺点
5. MongoDB 几大存储引擎
6. 数据复制的过程
7. 索引类型
8. 性能优化
9. 性能监控
10. GEO 索引算法

6. Spring Boot 2.5 实战 Redis 分布式缓存 6.0

内容简介：

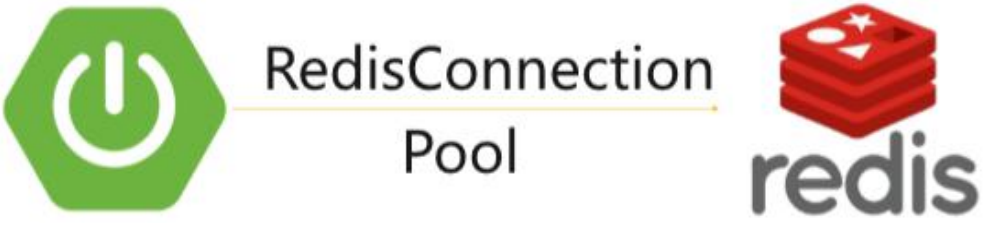
一、Spring Boot 2.5.x 实 战分布式缓存 Redis 6.0

二、Spring Data for Redis 架构

三、Linux Install Redis

一、Spring Boot 2.5.x 实战分布式缓存 Redis 6.0

1. Spring Data Redis6.0



Spring boot 项目集成是非常重要的缓存技术，头部及新兴的互联网公司，大量使用开源缓存非常普遍，本身 Redis 免费，且功能特性越来越完善，无论在基础功能使用还是丰富的数据类型、高级功能、主从集群模式、高可用集群以及分辨集群等方面的知识都非常好，现在也在扩展消息推送以及分布式事务等新的特性知识。

Redis 整个生态相对来说比较完善，无论是 Java 语言还是其他编程语言，另外比较重要的 Java 驱动、工具驱动库，有 Redis 的链接池，使用的是比较有名的 Java 社群 Jedis。

Spring Boot 为后续的 Java 应用开发作铺垫，Spring data for Redis 系列接口做了抽象，本质上 Java 连 Redis 有基础 Redis 的驱动，使用基础的网络链接和 Redis 服务进行交互，Redis 使用单机点模式，生产环境一般使用主重或高可用或正面集群模式，可以加设一台虚拟机编译安装完成。Java 链接远程 Redis，Redis 服务器端要允许远程端口链接，生产环境下请求安全验证。

2. Java Spring Data 2.x for Redis 新特性

1）支持多种 Redis 驱动程序/连接器的低级抽象（Jedis 和 Lettuce。JRedis 和 SRP 过期）

2）Spring Data Access exception 和 Redis driver exceptions 转换

3）RedisTemplate 高级抽象封装 Redis 操作，异常转换和序列化工作

4）Pubsub 发布订阅模式支持（例如消息驱动 POJO 的 MessageListenerContainer）

5）支持 Redis Sentinel 和 Redis Cluster 集群模式

6）JDK，String，JSON 和 Spring Object / XML 映射序列化器

7）基于 Redis 的 JDK Collection 实现

8）Atomic counter 原子计数器

9）Sorting and Pipelining 功能

10）专门 API 支持 SORT，SORT / GET 模式和返回批量值数据

11）Redis 实现了 Spring 3.1 缓存抽象

12）自动实现 Repository 接口，@EnableRedisRepositories 支持自定义查找方法

13）支持存储库的 CDI

Redis 本身也在不断迭代，功能越来越完善，Java Spring boot 连接使用 Spring Data for Redis，Redis 整个配置构建可以在替换基础的链接池组件，可以用 Jedis 和 Lettuce。

Redis 本身有单点也有集群模式，配置文件要注意配置参数的修改，整个 Redis 特性在 Java 原理使用中要考虑链接库的版本能支持这些操作。

Java 一定要用 Java8 或者 Jdk1.8 版本，Lettuce 也是 5.0 以后的版本，后面采用这个默认的集成模式。

过程中还比较有意思,实际 Redis 链接工厂，Factory 是属于工厂模式造链接词链接，每次创建 Redis 链接的时候，可以专门通过工厂类型的来进行创建，然后进行使用，另外可以通过配置给 Redis 客户端工厂添加必要配置。

3. Redis API

基础核心包 org.springframework.data.redis.core，里面包含两个重要类型，一、Redis Connection，二、Redis Connection Factory Interfac, Redis 链接对象需要指定 Redis 链接工厂，不同的链接池重构了 Redis 的链接工厂，接受的参数大部分一样，如主机、数据库、密码等重要参数都可控，同时也有默认策略，与链接 MySQL 的链接池

相像。

4. RedisConnection 解析

1）RedisConnection 为 Redis 通信提供核心组件

2）处理与 Redis 服务器后端的通信

3）自动将底层连接异常转换为 Spring DAO 异常

4）可以在不更改任何代码的情况下切换连接器

5）操作语义保持不变。

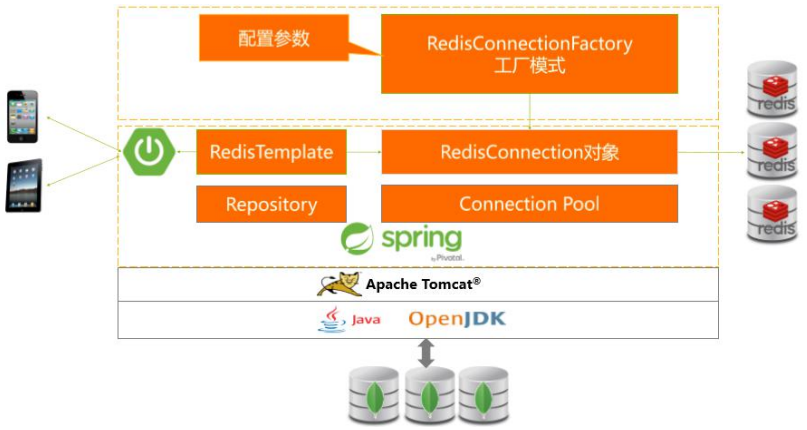
6）统一接口

7）工厂模式

8）仓储模式

二、Spring Data for Redis 架构

1. Java Spring Data for Redis 架构



2. RedisTemplate

Interface	Description
GeoOperations	Redis geospatial 操作, 例如 GEOADD,GEORADIUS,…
HashOperations	Redis hash 操作
HyperLogLogOperations	Redis HyperLogLog 操作,例如 PFADD, PFCOUNT,…
ListOperations	Redis list 操作 SetOperations Redis set 操作
ValueOperations	Redis string (or value)操作
ZSetOperations	Redis zset (or sorted set)操作

三、Linux Install Redis

1. Linux 安装 Redis 6.2

1）下载安装

- $ wget https://download.redis.io/releases/redis-6.2.1.tar.gz
- $ tar xzf redis-6.2.1.tar.gz
- $ cd redis-6.2.1
- $ make

2）启动服务器

- $ src/redis-server

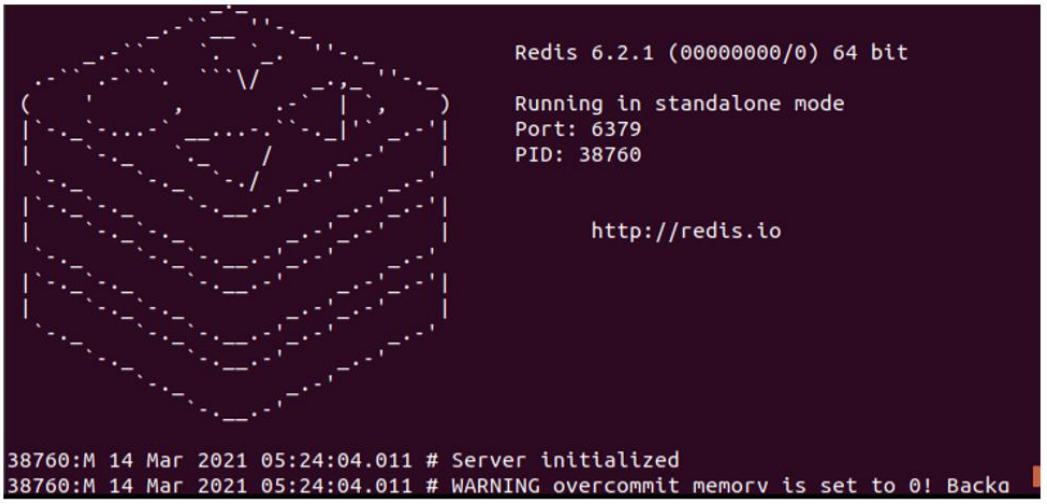
3）启动命令客户端

- $ src/redis-cli

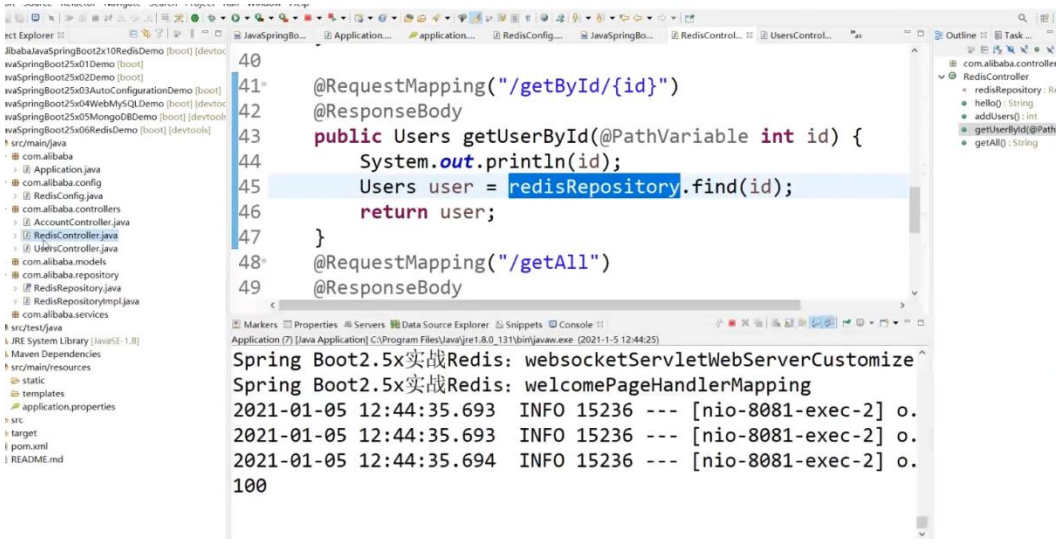
4）新增查询 Key value

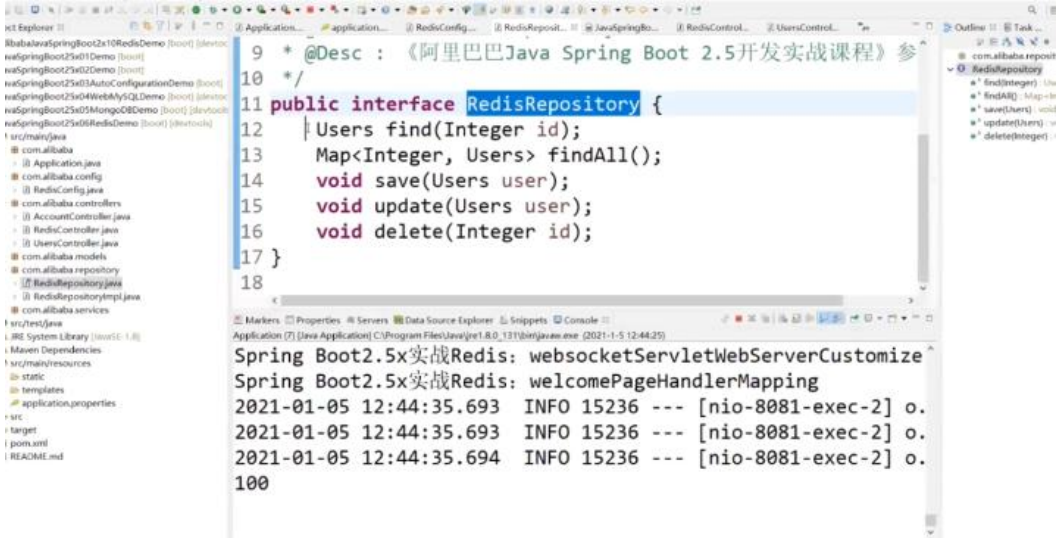
- redis> set 1 java
- OK
- redis> get 1
- “java"

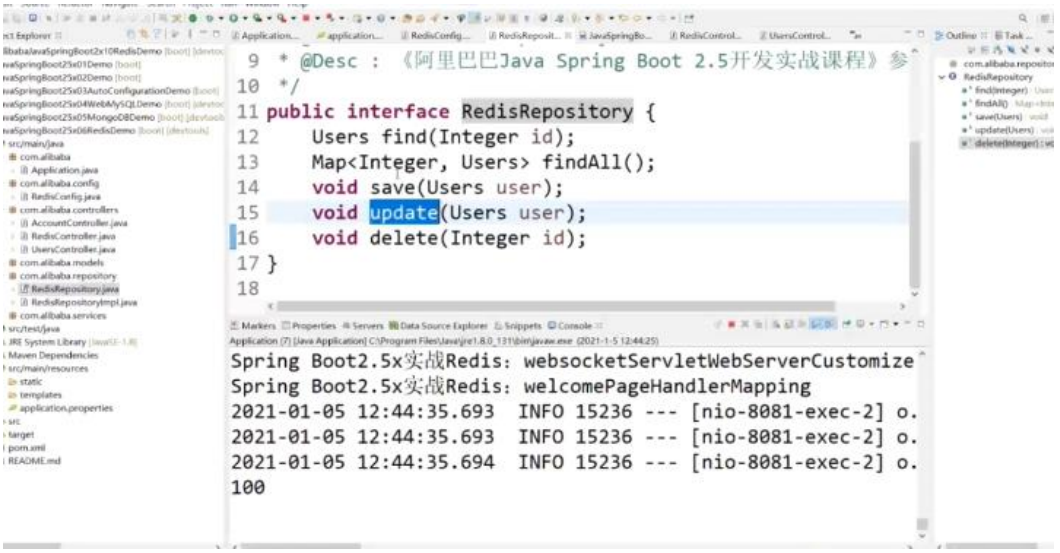
2. Linux 启动 Redis Server

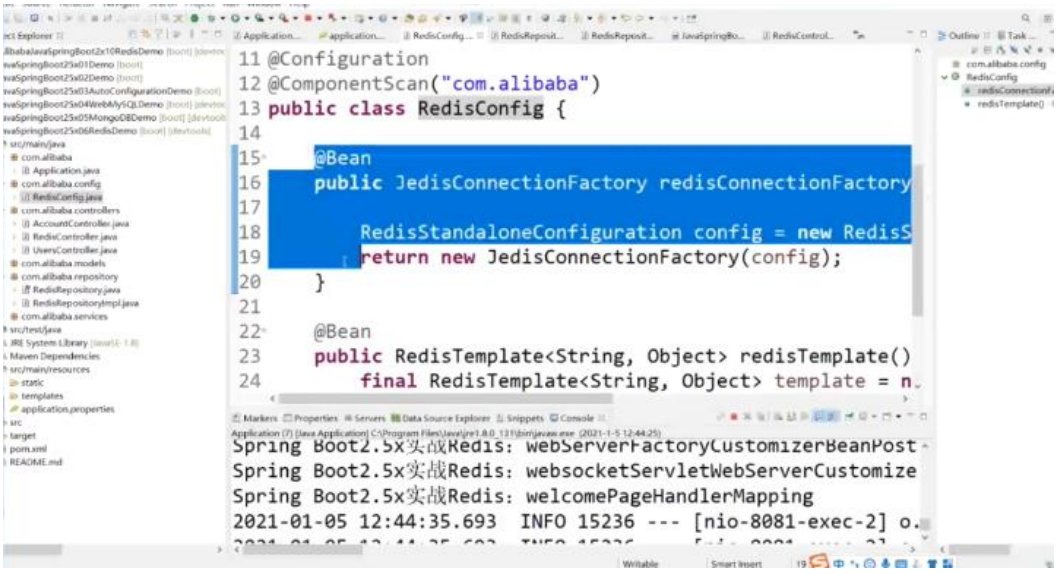


Redis Reactive Driver


@RequestMapping("/getById/{id}")
@ResponseBody
public Users getUserById(@PathVariable int id) {
System.out.println(id);
Users user = redisRepository.find(id);
return user;
}
@RequestMapping("/getAll")
@ResponseBody
Spring Boot2.5x实战Redis: websocketServletWebServerCustomize
Spring Boot2.5x实战Redis: welcomePageHandlerMapping
2021-01-05 12:44:35.693 INFO 15236 --- [nio-8081-exec-2] o.
2021-01-05 12:44:35.693 INFO 15236 --- [nio-8081-exec-2] o.
2021-01-05 12:44:35.694 INFO 15236 --- [nio-8081-exec-2] o.
100

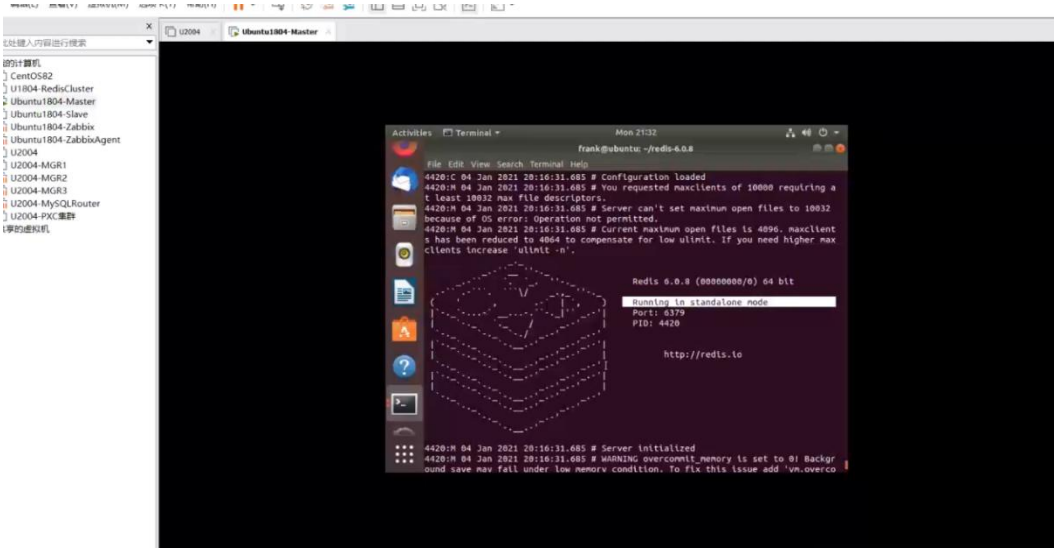

* @Desc : 《阿里巴巴Java Spring Boot 2.5开发实战课程》参
*/
public interface RedisRepository {
Users find(Integer id);
Map<Integer, Users> findAll();
void save(Users user);
void update(Users user);
void delete(Integer id);
}
Spring Boot2.5x实战Redis: websocketServletWebServerCustomize
Spring Boot2.5x实战Redis: welcomePageHandlerMapping
2021-01-05 12:44:35.693 INFO 15236 --- [nio-8081-exec-2] o.
2021-01-05 12:44:35.693 INFO 15236 --- [nio-8081-exec-2] o.
2021-01-05 12:44:35.694 INFO 15236 --- [nio-8081-exec-2] o.
100

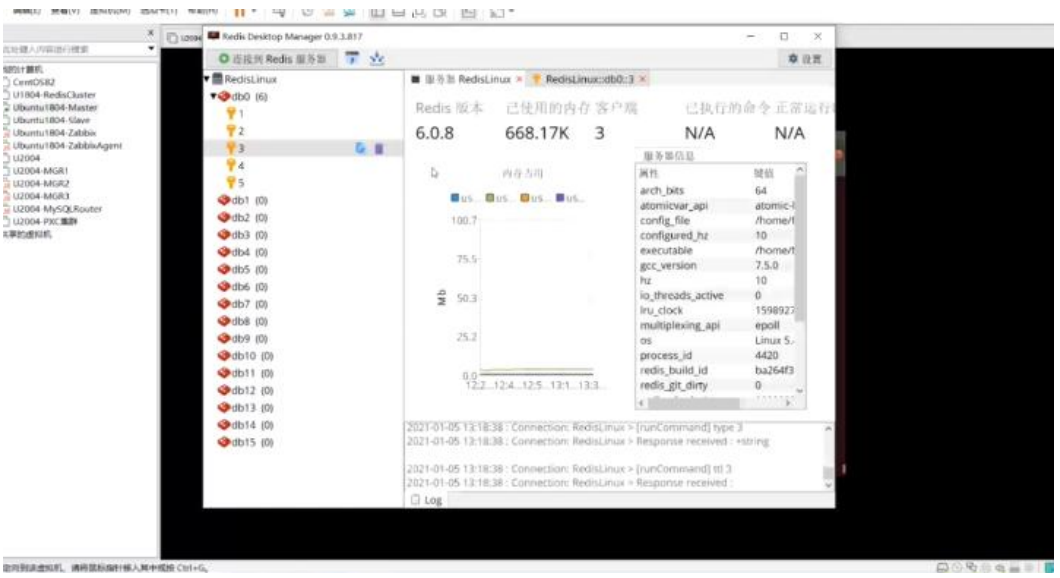


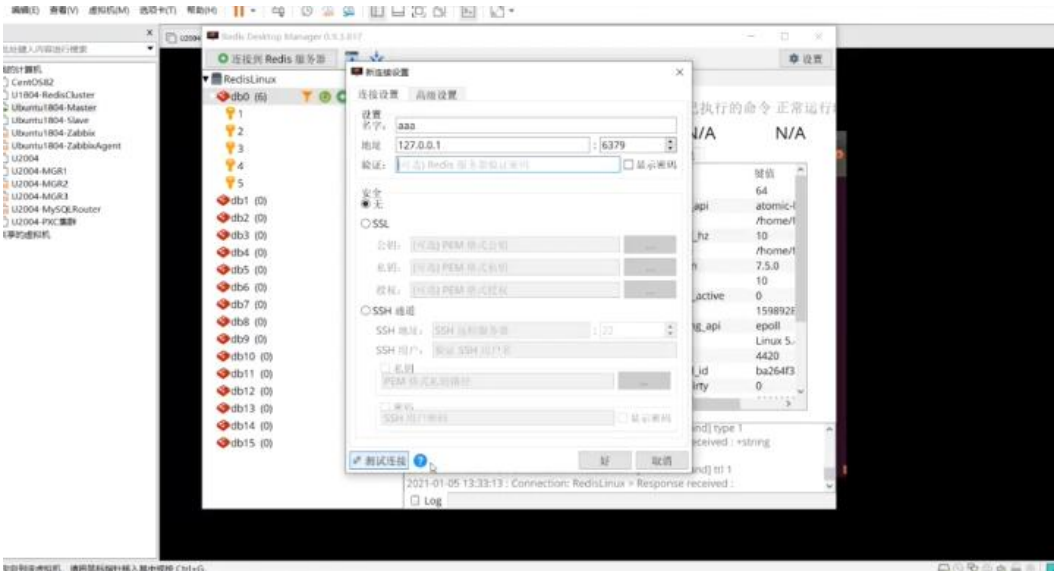


如上图所示，与 MangoDB、MySQL 差不多，加了一个控制器，里面实际放的一个是新增，一个是查询，查询调的是 redis 仓储类型，里面依赖注入，接口是自定义的一些方法，实现加了 redisTemplate 类型，用这个模板类型做 redis 对象操作，主要做基础的增删改查，基本操作用较新的默认版本通过依赖加了 jedis，Spring Boot2.5 以后会

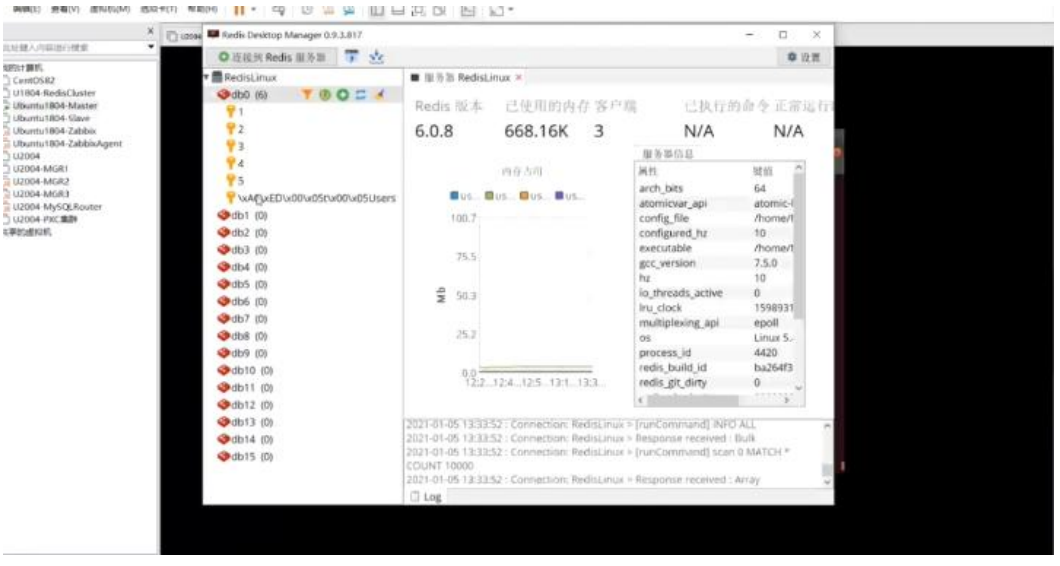
自动版本兼容，自动拉数据有 page，Redis 对应的一些参数可以从 page 文件里进行数据读取，也可以写死， bean 的构造方法可以指定传主机名、传端口，这两个参数通过 value 注解的方式配置文件自动加载完成。







如上图所示，启动完以后要验证整个项目，这里是 6.0.8 版本，属于单点独立模式启动，只有一个节点是为了方便调整，如果做远程链接， Ip 保护模式限制要解除，需要修改 IP，如果不喜欢用命令行可以用可视化工具。



如上图所示测试， Ad user 里保存多少，调用接口，保存 100 个用户，代码控制器跟之前差不多，get by ID ， add user for 循环，构造 100 个用户，插入 list，后面 get ALL 实际调 repository 的仓储对象做查询操作。

注意第一步 Activities 先装，启动后配置网络和安全模式，必须远程链接客户端才能够进行链接操作。

7. Spring Boot2.5 安全机制与 REST API 身份验证实战

内容简介：

一、Java Spring Boot 2.5 安全机制

二、Java Spring Boot 2.5 安全实战

一、Java Spring Boot 2.5 安全机制

本节课讲的是应用程序安全问题，在 Spring Boot 体系里，提供了一套安全机制，可以对接各种不同的安全框架，包括自定义实现原始的身份验证机制。

模拟简单的 REST API 项目，启用身份验证，进行扩展对接 MySQL 数据库，甚至对接 Release 缓存，实现整个用户的注册和身份验证过程。

但大型项目，比如淘宝、微信、新浪微博的账号验证基本上都在 Release 里面进行，用户的规模比较大，而且整个的用户会话信息，要在分布式缓存里面进行保存。后面可以扩展到 Release 身份验证机制，把身份验证与 Release 和 MySQL 结合起来，实现一个完整的项目。

1. Java Spring Boot 2.0 安全机制

Spring Boot 提供的安全机制，可以用 Spring Security 开源框架，也可以用 Apache Shiro 开源框架，还可以用自定义实现安全验证。Web 框架开发底层本质上是 Web 请求进入 Web 框架，然后可以拦截，这里也叫 AOP 编程，用于做拦截，做身份验证的工作。总结：

1）自定义实现安全验证；

2）Apache Shiro 开源框架；

3）Spring Security 开源框架；

4）大量使用 AOP；

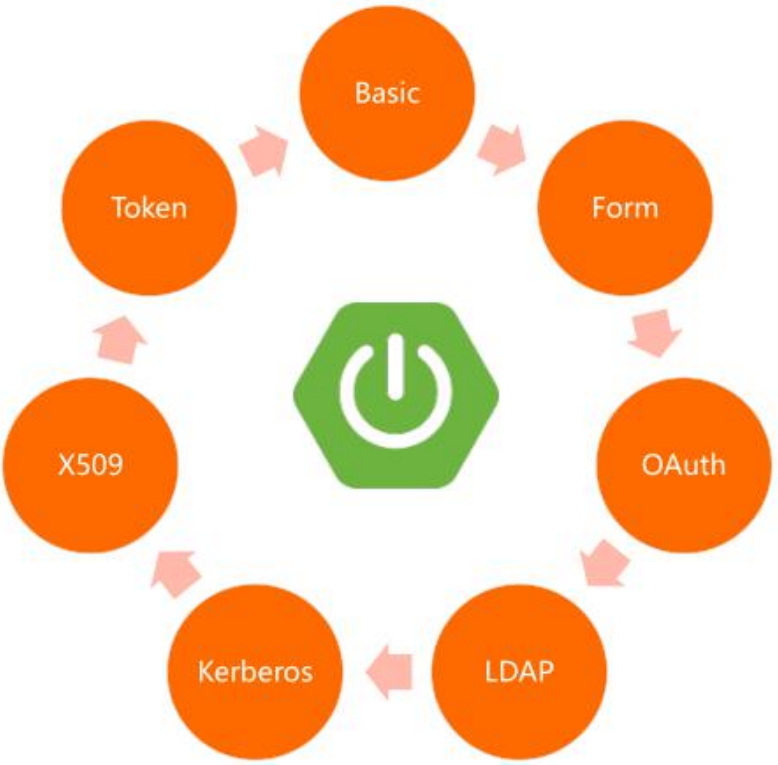
5）依赖注入思想；

6）灵活扩展。

2. Java Spring Boot 2.5 安全机制

也可以进行授权，定了一些角色，设置对应的权限，这里支持的方式很多。Spring Boot 作为快速开发框架，底层有 Web 开发的接口，可以做网站、 API 应用、定时任务应用等等。

目前应用程序身份验证的类型非常多，不仅基于网页的 Form 表单验证、Basic、摘要身份验证、令牌身份验证，令牌身份验证主要是用在 REST API，包括一些大型的微服务架构程序中。当然还有企业级身份验证，比如基于企业局域网的客户身份验证，组织内部使用的身份验证。还有跨第三方平台的开放式身份验证体系等。



3. 安全漏洞

后续基于 Spring Boot 进行开发，尽量升级到比较新的版本，最好是 Spring Boot 2.3 以上的版本，不要用太老的版本。Spring Boot 2020 年 9 月份又修复了一批安全漏洞，涉及到远程代码过程执行的安全漏洞问题。

安全漏洞建议总结：

1）Spring Boot 2020 年 9 月份修复漏洞；

2）Spring Boot Actuator 未授权访问远程代码执行漏洞；

3）紧急修复 Spring Framework 版本包含一个安全漏洞（CVE-2020-5421）的修复程序。此漏洞可以通过 sessionId 绕过 RFD （反射型文件下载）保护；

4）Spring Boot 2018 年修复了一些安全漏洞；

5）建议使用最新的 Spring 5.0+版本；

6）Spring 框架升级 5.0.0 - 5.0.2；

7）Spring 框架升级 4.3.0 - 4.3.13；

8）Spring Boot 1.5.10。

4. Java Spring Security

安全框架本身相对成熟，可以集成 Spring Security，也可以集成 Spring MVC，也可以集成 Spring Boot，也可以集成 Spring cloud。

保护 Spring 应用系统的安全标准，可以实施各种产同的身份验证，可以做各种数据源进行集成，定制开发工作，包括基于角色、基于令牌都可以实现。这种多种身份验证机制的支持，包括扩展 URL、自定义路由规则，这种方法的验证都可以进行实现。这种

框架方便构建安全体系，不会限制应用程序的类型。整个集成和 Spring 应用平台集成的最好。**总结：**

1）Spring Security 功能强大且高度可自定义的安全框架；

2）保护 Spring 应用系统的安全标准；

3）Spring Security 专注于身份验证和授权；

4）容易地扩展、自定义开发；

5）前身是 Acegi Security；

6）提供安全认证服务的框架；

7）Spring Security 为基于 J2EE 企业应用提供了全面安全机制；

8）Authentication 验证和 Authorization 授权；

9）抵御会话攻击，点击劫持,CSRF 跨站请求伪造。

5. Java 安全框架 Shiro

Apache 官网有一个开源的项目叫 Apache Shiro，是一个开源的安全管理，支持的工作非常强大，唯一的差别是不属于 Spring 官方。但是很好用，有很多项目包括 Spring Boot，甚至 Spring Cloud 都在使用 Apache Shiro。对接不同的数据源，两个都可以，取决于个人倾向，两个都非常完善，正常情况下，如果希望技术上简单一点，可以直接使用一整套 Spring。Java 安全框架 Shiro 总结：

1）Apache Shiro 简单易用的开源 Java 安全框架；

2）轻松实现身份验证、授权、加密和会话管理；

3）使用 Shiro 可以快速实现系统安全；

4）Shiro 其前身是 Jsecurity 项目；

5）Shiro 可以轻易实现 Java 网站安全验证；

6）可应用于 Web 环境，非 Web 环境；

7）支持多种数据源 MySQL 等；

8）如 LDAP，JDBC，Kerberos，ActiveDirectory 等）。

二、Java Spring Boot 2.5 安全实战

1. Spring Security Demo

在之前的 WEB 架构基本上，可以加入门的 starter-security 的依赖，入完以后，会提供必要的安全组件，默认的就是 security 组件，当然也可以替换，两个都可以配置。实现代码如下:

```
<dependencies>
 ...
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId> spring-boot-starter-security </artifactId>
</dependency>
 ...
</dependencies>
```

2. WebSecurityConfig

早期的 Spring1.x 几版本中，需要自己显示去关闭或者开放安全验证，现在的基本上默认是已经启动了。可以直接设置 WebSecurityConfig 配置文件，注入必要的安全配置规则，比如基于某个内路由规则、针对 order 订单的身份验证等，甚至加入角色的限制都可以实现，提供了强大的规则。

WebSecurityConfig 总结：

1）Spring Security 的配置类；

2）WebSecurityConfig；

3）可以配置安全规则；

4）默认启用 basic 验证；

5）# Spring Security 可以在配置文件中关闭；

6）security.basic.enabled = false。

3. Web 全站安全验证配置

下面这段代码，是 Web 全站安全验证配置的例子，针对 anyRequest 定制时，所有的请求都做身份验证。hasRole 表示针对当前的请求访问某个地址，或者所有的请求访问时是否具备某个角色。当然这个角色或者权限的验证，需要定制扩展接口就可以了。比如连数据库或者连缓存，实现角色查询，有校验就可以了。

```
@ Configuration
@ (SecurityProperties.BASIC_AUTH_ORDER - 10)
public class ApplicationConfigurerAdapter extends
WebSecurityConfiguerAdapter{
```

```
@Override
protected void configure( http) throws
{
http.antMatcher("/admin/**")
.authorizeRequests()
.antMatchers("/admin/users").hasRole(“usersAdmin")
.antMatchers("/admin/orders").hasRole(“ordersAdmin")
.anyRequest().isAuthenticated();
}
}
```

4. 重新实现验证机制

当然有一般规则，就有特殊规则，允许自定义扩展底层的 API 实现自定义的判断逻辑，这是良好的安全框架所具备的功能。重新实现验证机制：

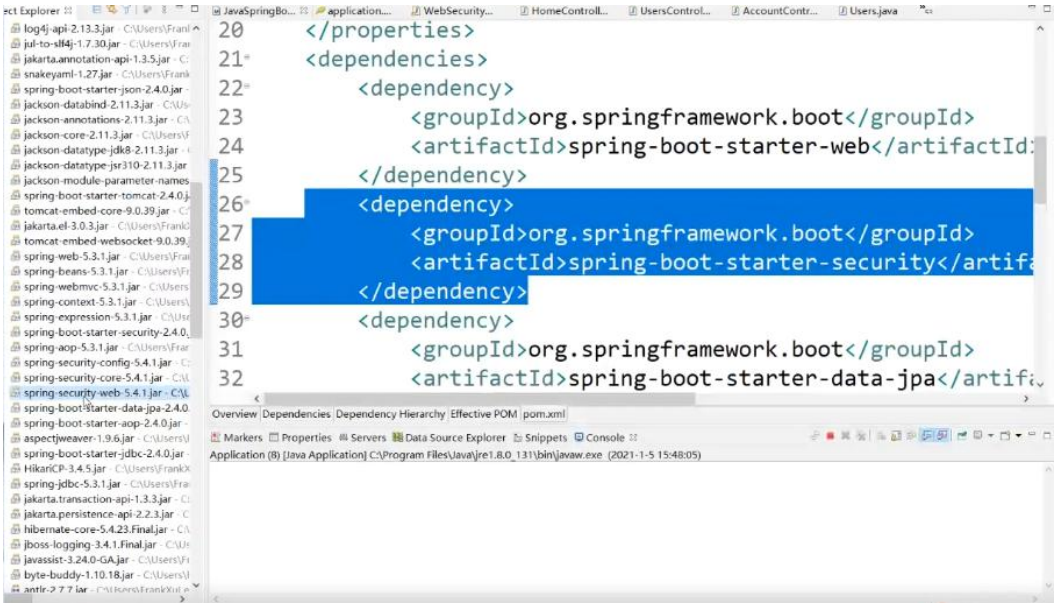
- WebSecurityConfig 安全配置类；
- UserDetailsService 接口。

实践案例演示:

这个程序是之前的 Web 网站，看一下项目的依赖，通过文件依赖可以看出，相比之前的项目，数据库依然存在，还是用之前的 spring data jpa,加入 MySQL 的数据库的链接。

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifa
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifa
</dependency>
<dependency>
    <groupId>mysql</groupId>
```

后面稍微注意一下，这里面有个 starter security，加完以后，可以看一下左边的依赖包的安全组件，有包括 spring security config 、spring security core、spring security web 等。



下面看一下，代码具体的配置和实现，新版本默认已经集成，现在最主要的要有 Web 安全配置，全局配置的时候使用内存验证，实际可以对接数据库、对接缓存。当前程序的进程里面写了用户名“frankxu”,密码“1234qwer”,这种方式并不提倡。

```
.inMemoryAuthentication()
    .withUser("frankxu")
    .password("1234qwer")
    .roles("ADMIN");
```

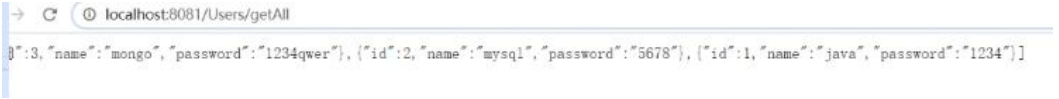
正常情况下，可以定义特殊规则“configuer”,针对 rize 请求，可以匹配首页，允许首页所有的人都可以访问；针对 user，做身份验证。也可以提供表单的登录模式“formLogin()”。做这个例子，是提供这种访问 HOME 可以不做身份验证的方法，可以直接进入。

```
.authorizeRequests()
    .antMatchers("/", "/Home").permitAll()
    .antMatchers("/Users/**")
    .access("hasRole('ROLE_ADMIN')")
    .anyRequest().authenticated()
    .and()
.formLogin();
```

查询数据库 Get all，演示在浏览器输入“localhost:8081”,启动网址，会发现一个问题，访问一个接口时，比如 user 查看所有用户，会自动跳转到“login”。当请求一个地址的时候，发现不是可忽略的地址，会自动拦截，跳转到登录界面，这个登录界面是基于表单。

这时会要求输入用户账号和密码，用户账号和密码是定义死的。

为我之已经测试过了，点击登录，“Users/gerAll”就可以访问了，



如上图所示，出现了 3 个账号和密码，这 3 个账号全部返回出来了。当然这里还有别的接口，“GetById”

```
    }

    @PreAuthorize("hasRole('Admin')")
    @RequestMapping("/getAll")
    @ResponseBody
    public List<Users> getAllUsers() {
        List<Users> listUsers = (List<Users>) usersDAO.getA
        System.out.println(listUsers.size());
        return listUsers;
    }

    @RequestMapping("/getById/{id}")
    @ResponseBody
    public Users getUserById(@PathVariable int id) {
```

测试一下，输入“GetById/1”,返回 ID 为 1 的账号和密码:

五、Java 面试题

1.本质：URL，拦截请求，验证，放行或者拒绝

注意规则的顺序，正常的项目，会有一个访问，并且有拦截请求操作，安全机制本质上是拦截请求，基于 URL 规则，判断请求是不是要拦截，验证，然后放行或者拒绝。验证过程，可以调用缓存或数据库实现安全验证规则。

其他的 Java 面试题这节课就不一一说明了，留给大家课后思考。

2. Java Spring Security 安全机制;
3. 如何扩展使用 Token 令牌验证;
4. JWT 开源安全令牌组件;
5. 如何支持 X509 正式验证;
6. 自定义实现 Spring Boot 2.5.x 身份验证;
7. API 安全如何实现;
8. SSO 单点登录怎么实现?
9. 微服务 Spring Cloud 安全体系;
10. 转 Java，跳槽一线互联网公司。

8. Spring Boot 2.5 实战 API 帮助文档 Swagger

内容简介：

一、REST API 帮助文档

二、REST API 自动生成帮助文档 Swagger

一、REST API 帮助文档

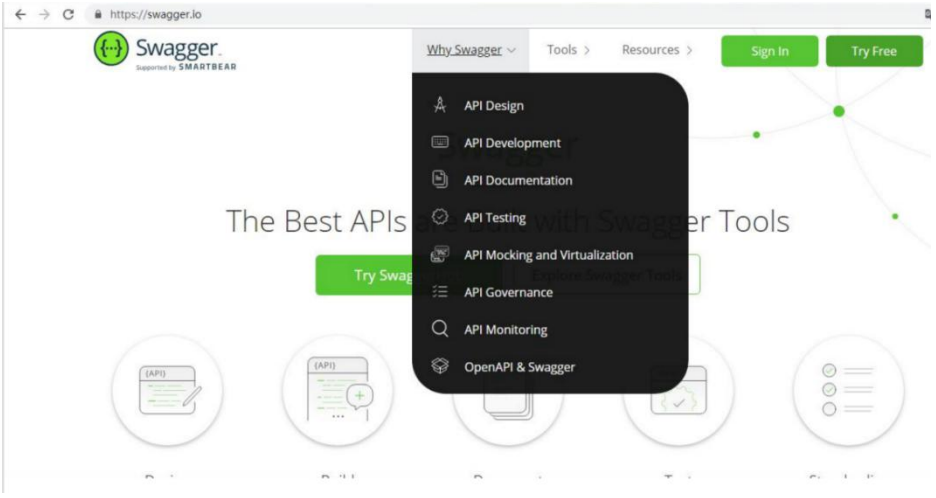
目前大型移动互联网平台，像淘宝、微信，抖音、拼多多，包括滴滴打车、美团等，都是前后端分离的架构，我们叫微服务架构。

Swagger 是自动化 API 文档的生成工具，这个工具之前是 Spring Boot 项目来进行集成，现在使用 Spring Boot 项目做后端开发，写 API 的代码的机会比较多，咱们把这个工具给大家介绍一下。

作为一个快速开发框架，是 Spring Boot 提供了自己的一套 API 的文档工具，目前来看，Swagger 使用比较多，这几年来普及率非常高，因为它非常方便，它的文档生成基本上都是自动化，只需要加一个简单的处理就可以。

对于后端开发来说，不需要自己专门写一套 word 文档，发给前端，前端再自己去测试，再调进来后台 API。Swagger 文档部署完以后，前端可以直接拿到，然后进行在线调试，非常方便。简化前后端协助，协助避免出错。

二、REST API 自动生成帮助文档 Swagger



官网 https://swagger.io

1. Swagger 自动化文档工具

1）Swagger 是一个完整的 API 生态，工具，规范，代码生成。

2）用于描述，生成，使用和可视化 RESTful Web 服务。

3）Swagger API project 2011 Tony Tam 创立 最早 Java 版。

4）SmartBear Software 公司支持，Apache License 2.0。

5）OpenAPI Spec。

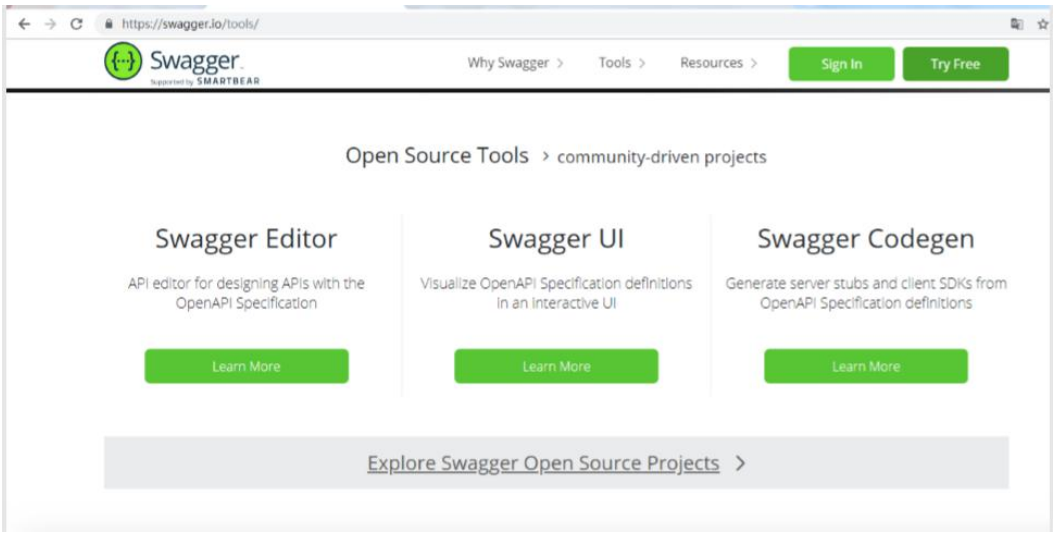
6）Swagger and OAS。

7）Swagger 2 to OpenAPI 3。

8）捐赠给 linux 基金会。

9）行业标准规范。

10）Swagger Tools 一套工具：设计、开发、测试、监控、治理。



2. Spring REST Docs

Spring REST Docs

1）Spring REST Docs 帮助自动化生成 RESTful 服务的文档。

2）使用 Asciidoctor 编写的手写文档。

3）Spring REST Docs 为 RESTful 服务生成准确且可读的文档。

4）将手写文档与使用 Spring 测试生成的文档片段相结合。

5）不受 Swagger 等工具生成的文档的限制。

6）它可以生成准确，简洁和结构良好的 API 文档。

7）Spring REST Docs 支持测试驱动 Test Driven。

8）pring REST Docs 支持 Spring MVC Test 框架，Spring WebFlux 的 WebTest Client 或 REST Assured 3 测试驱动。

9）Spring Boot 提供了注解@AutoConfigureRestDocs。

10）替代 SpringFox Swagger。

3. 优点

1）手写文档与使用 Spring Test 框架生成的文档片段结合。

2）curl and http request snippets are generated。

3）easy to package documentation in projects jar file。

4）easy to add extra information to the snippets。

5）supports both JSON and XML。

4. MockMvc

1）MockMvc 是 Spring MVC Test 工具类，支持 Assert 和 Chain。

2）@Mock 创建模拟对象，Mock。

3）@InjectMocks 会自动将 mock 依赖注入测试对象。

4）MockitoAnnotations.initMocks（this）初始化。

5）MockMvcBuilders.standaloneSetup（..）.build（）通过注册一个或多个@Controller 实例并以编程方式配置 Spring MVC 基础结构来构建 MockMvc 实例。

6）@Test 标注测试方法。

7）@WebMvcTest 注解用于 Spring MVC 测试。 它禁用完全自动配置，而只应用与 MVC 测试相关的配置。

8）WebMvcTest 注解也自动配置 MockMvc 实例。

5. Asciidoctor 插件步骤

1）pom.xml 添加 Asciidoctor 插件

2）添加对 spring-restdocs-mockmvc 的依赖

3）配置属性 asciidocs 输出位置 sourceDirectory

4）配置测试任务 task 输出位置 outputDirectory

5）配置 asciidoctor task

6）snippets 定义 snippets 输出位置

7）使 task 依赖于 test 任务，以便在创建文档之前运行测试

8）将 snippets 配置为输入。将在此目录下创建所有代码段

6. REST Assured

1）Rest-Assured 由 Java 实现的 REST API 测试框架。

2）在 Java 中测试和验证 REST 服务比在 Ruby 和 Groovy 等动态语言中更难。

3）REST Assured 简化 REST API 测试。

4）专为测试 REST API 而设计的 DSL。

5）Java DSL，用于轻松测试 REST 服务。

6）REST Assured 支持任何 HTTP 方法，但明确支持 POST，GET，PUT，DELETE，OPTIONS，PATCH 和 HEAD，并包括指定和验证例如 parameters, headers, cookies body 。

7）自动化测试

8）http://rest-assured.io/

7. Spring Auto Rest Docs

Spring REST Docs 最低要求

1）Java 8

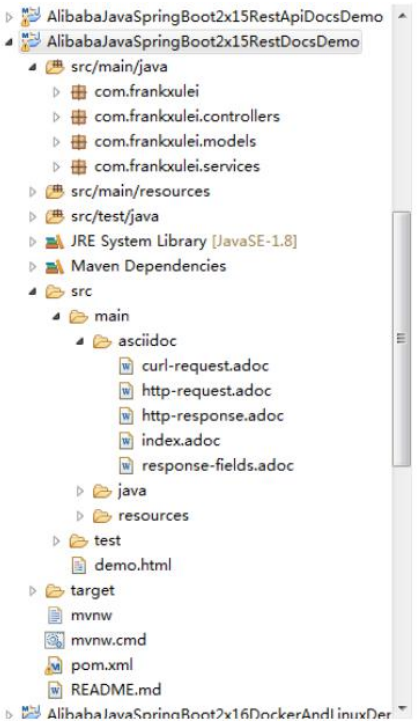
2）Spring Framework 5 (5.0.2 or later)

3）此外, the spring-restdocs-restassured 要求 :

4）REST Assured 3.0

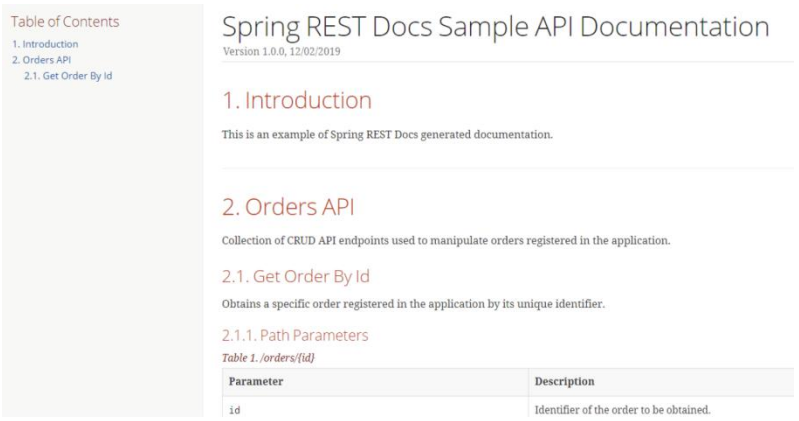
8. Spring Rest Docs Demo

Spring Rest Docs 实战



9. Asciidocs Maven Plugins

```
<plugin>
                <groupId>org.asciidoctor</groupId>
                <artifactId>asciidoctor-maven-plugin</artifactId>
                <version>1.5.7</version>
                <executions>
                    <execution>
                        <id>generate-docs</id>
                        <phase>prepare-package</phase>
                        <goals>
                            <goal>process-asciidoc</goal>
                        </goals>
                        <configuration>
                            <backend>html</backend>
                            <doctype>book</doctype>
                        </configuration>
                    </execution>
                </executions>
                …
</plugin>
```



Spring REST Docs可以在线方便的调试自己的API,但是没有 Swagger 使用方便，这边就简单介绍下，重点还是讲实战 Swagger。

10. Spring Boot 2.0 实战 Swagger

```
<dependency>
<groupId>io.springfox</groupId>
<artifactId>springfox-swagger2</artifactId>
<version>2.9.2</version>
</dependency>
<dependency>
<groupId>io.springfox</groupId>
<artifactId>springfox-swagger-
ui</artifactId>
<version>2.9.2</version>
</dependency>
```

引用 Swagger 的包，需要自己做一些参数化的配置，简单的可以在配置文件进行，复杂一些配置需要在代码里面进行。生成的调试方式也比较简单，生成的网页里面在详细的检索描述性，可以在线的发送 get、Post 等经典请求格式，很方便的去调接口，对于前后端分离的架构来说是很方便。

页面打开两种方式：

http://localhost:8081/swagger-ui.html

1）/v2/api-docs

2）Swagger UI /swagger-ui.html

接口文档的版本可以不断的变化，也可以在后台进行配置。

11. Swagger-core 注解

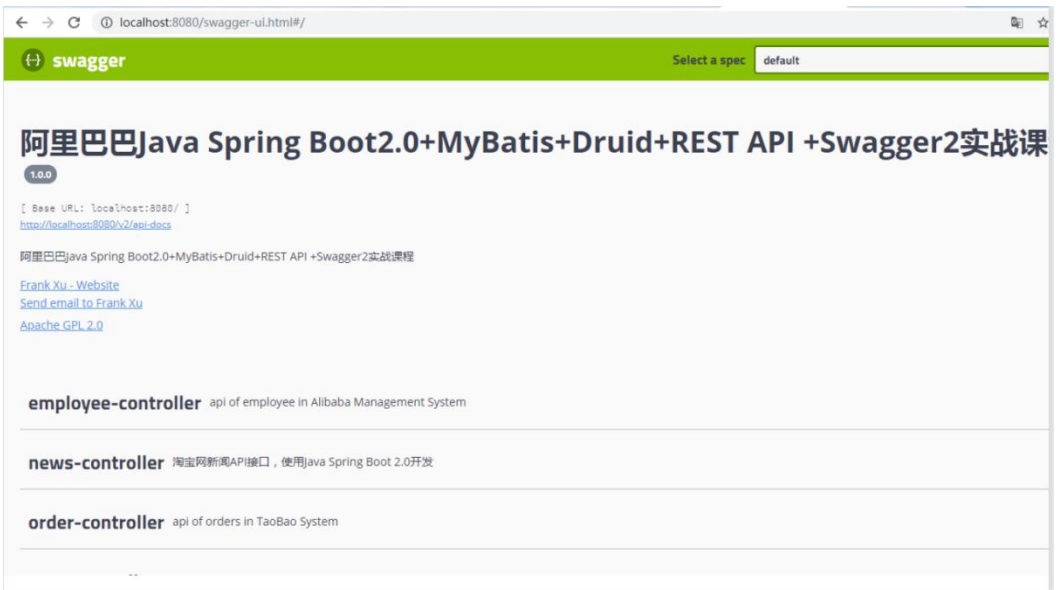
名称	描述
@Api	Marks a class as a Swagger resource.
@ApiModel	Provides additional information about Swagger models.
@ApiModelProperty	Adds and manipulates data of a model property.
@ApiOperation	Describes an operation or typically an HTTP method against a specific path.
@ApiParam	Adds additional meta-data for operation parameters.
@ApiResponse	Describes a possible response of an operation.
@ApiResponses	A wrapper to allow a list of multiple ApiResponse objects.

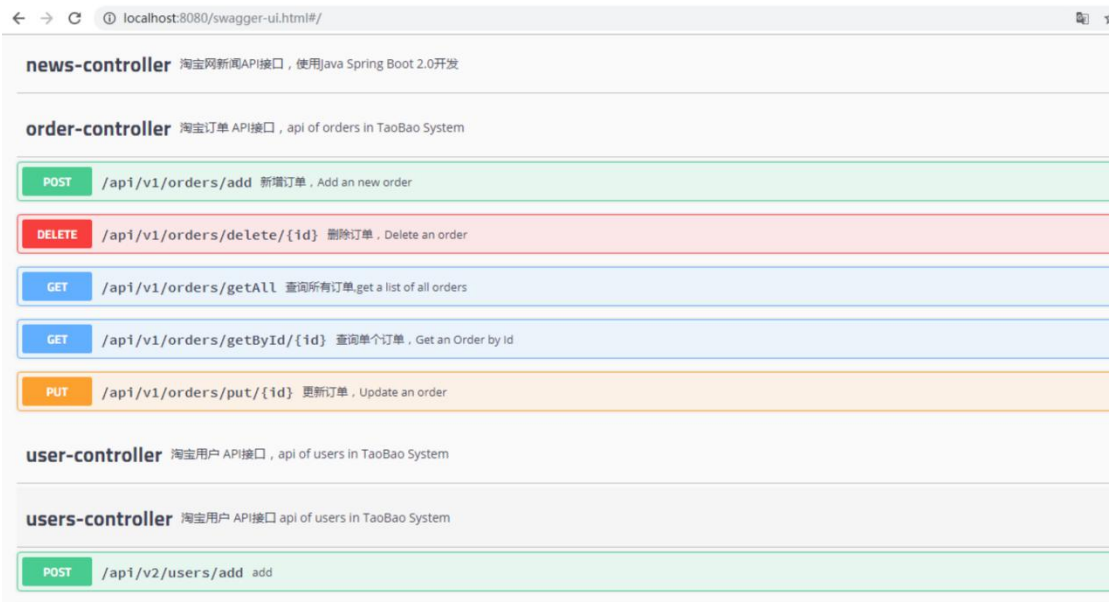
在开发过程中，默认的话什么都不加的话，实际解析的信息如说控制器或者类别的基本信息。如果希望对内加一些描述信息。对原接口加原表述信息的话，可以加进来如传输的数据类型，加个 model 的 API model 加个说明，模组里面字段你可以加 property 这个说明。操作具体方法的话可以 operation，参数的话有 pyramid 的说明，应答消息和请求消息的话也可以加 response，这种相对的这些注解说明都可以了。它会自动的把这些信息提取出来，生成放到 Swagger 在线帮助文档里。

12. Spring Boot 2.0 Rest API 注解

```
34
35    @ApiOperation(value = "View a list of available users", response = List.class)
36    @ApiResponses(value = { @ApiResponse(code = 200, message = "Successfully retrieved
37            @ApiResponse(code = 401, message = "You are not authorized to view the res
38            @ApiResponse(code = 403, message = "Accessing the resource you were trying
39            @ApiResponse(code = 404, message = "The resource you were trying to reach
40    @GetMapping("/getAll")
41    public List<Order> getAll() {
42        return orderRepository.getAll();
43    }
44
45    @ApiOperation(value = "Get an Order by Id")
46    @GetMapping("/getById/{id}")
47    public ResponseEntity<Order> getById(
48            @ApiParam(value = "Order id from which Order object will retrieve", requir
49        Order order = orderRepository.getById(id);
50
51        return ResponseEntity.ok().body(order);
52    }
```

应答消息 401、404、403 等消息可以自己定制，如整合 API 的类型的话，可以加入淘宝用户的 API 接口等，根据自己的需求进行添加。





接口也可以分类，目前是这里简单做了几个分类：订单接口，用户接口，并可以在里面进行测试，方便在线检查，并视图形式，反馈各个消息类型的结果。

测试有错注意点：

- 需要使用 RestController，不要是用 Controller。
- 出现 order repository 问题，是没有数据的原因。
- Swagger 功能非常强大，也方便调试开发，尤其是前后端分离的架构。下节课继续讲 Spring Boot 的性能监控，内容容器等重要内容。

9. Spring Boot2.5 实战 – 应用程序性能监控

内容简介：

一、Spring Boot 实战 – 应用程序性能监控 APM

二、Spring Boot 2.4 实战 –内置性能监控 Endpoint

三、Java Spring Boot 2.5 监控实战

欢迎收看 spring boat 实战开发系列课程，这节课也讲另外一个非常重要的主题叫应用程序的性能监控。在生产环境下，应用程序的性能监控是非常重要的，开发阶段可能还不严重，但是生态环境下的话有可能基于高并发，比如双 11 等带来的这些流量冲击问题，可能应用程序在生产环境下，会因为一些配置参数等网络问题，导致应用程序性能可能出现这个问题，比如在高并发等会导致大量的内存耗尽等问题，或者应用程序代码本身在前期开发设计阶段做得不够详细，会出现缓慢的内存泄漏，都需要去做应用程序的性能监控，保证应用程序的强壮性，以及我们后期的维护等。

一、Spring Boot 实战 – 应用程序性能监控 APM

1. Spring Boot 2.0 监控指标

基于 Spring Boot 应用程序，我们如何做应用程序的性能监控。对于微服务架构的

市场开发课程，也是很重要的一个铺垫，大家需要实战操作去理解我们整个课程内容，包括我们会讲非常重要的里面提供了一个组件，executed 还有社区里面一个重要的开发项目，Spring Boot2.5，来帮我们去提供可视化的一套接口，Spring Boot 本身也提供这种对应的一些接口，你可以去抓取这些原始的数据，我们在后续使用 Spring Boot 进行开发的话，这些信息也都可以抓得到，只不过我们需要大家去了解，结合自己的整个监控需求，进行配置相应的接口。

2. Spring Boot 性能监控 Actuator

1）Spring Boot 2014 年 4 月性能监控和管理组件 Actuator

2）使用 HTTP Endpoint 或 JMX，运行状态指标数据收集

3）health, metrics, info, dump, env, etc

4）2.0 以后改进监控地址带有/actuator 前缀

5）例如/actuator/health 监控健康状态信息

6）禁用 management.endpoint.shutdown.enabled=true

7）单个禁用

8）management.endpoints.enabled-by-default=false

9）management.endpoint.info.enabled=true

Actuator，主要是来帮助你去进行应用程序的一些监控参数的收集，并且进行扩展暴露给第三方程序的。地质列表里面实际针对的各种不同的一个数据的暴露的信息，这里面比较重要的应用程序性能监控的这些数据都属于敏感数据，不能轻易的暴露给第三方及外界。当然还有另外一个问题就是这是一把双刃剑，它本身的性能监控指标的数据收集的话，一定也会消耗一定的服务器资源，比如我们讲的是 CPU 或者我们说内存，这里面还会通过网络暴露的话会消耗一定的网络带宽资源，它提供了各种不同的这种监控指标信息的话，实际都属于我们说叫 Endpoint 的系列都给你提供了一个暴露的端点或者叫终结点。

二、Spring Boot 2.4 实战 –内置性能监控

1. Endpoint

Endpoint列表

阿里云

内置EndPoint ID	描述	默认启用
auditevents	暴露审计事件信息	Yes
beans	Beans列表	Yes
caches	缓存信息	Yes
conditions	评估配置的条件信息	Yes
configprops	显示 @ConfigurationProperties列表	Yes
env	环境信息	Yes
flyway	Flyway 数据掐你一信息	Yes
health	健康信息	Yes
httptrace	显示HTTP 跟踪信息(默认最近100 HTTP请求应答).	Yes
info	显示程序信息	Yes
integrationgraph	集成图Spring Integration graph.	Yes
loggers	显示与修改日志信息	Yes
liquibase	Liquibase 数据库迁移信息	Yes
metrics	显示‘metrics’ 指标信息	Yes
mappings	显示@RequestMapping路径列表	Yes
scheduledtasks	显示调度任务	Yes
sessions	查询和删除会话Spring Session-backed session store. 不支持reactive web applications.	Yes
shutdown	优雅地关闭应用	No
threaddump	执行线程Dump thread dump.	Yes

通过网络给你提供了一个数据暴露的接口，现在就可以用第三方的一些展示数据或自己来开发，开发一个 web 界面来去展示。

Web

阿里云

EndPoint ID	描述	默认开启
heapdump	返回hprof heap dump文件.	Yes
jolokia	使用HTTP 协议暴露JMX beans (当Jolokia位于classpath, 不支持WebFlux).	Yes
logfile	返回logfile(如果配置了logging.file或 logging.path).支持使用HTTP Range标头来检索日志文件内容。	Yes
prometheus	以可以由Prometheus服务器抓取的格式公开指标数据	Yes

1、web 对接可以抓取原始的单位信息，可以通过 HTTP 协议或者我们 JMX 来暴露可以对接 prometheus 原生环境下比较重要的一个心理监控的一套解决方案。

例子：2.0 以后启动了一个我们说的这样一个应用程序，这个地址上下面有很多子目录，对应的这些不同信息的地址，然后你可以去抓取他然后进行展示，是非常重要的。

2、2.0 默认监控 EndPoint

后面看如何和实际项目进行结合，讲了很多理论知识和概念，也是希望告诉大家这里面对于我们的后续大家开发使用 Spring Boot 应用程序，包括 recipe 项目前后端分离的架构的项目，包括我们说微服务架构项目集群，这都是非常重要的一个方式，而且我们整个微架构进行监控的话，也是在这个基础上进行了升级改造。

你如果要做应用程序的性能监控，这是敏感数据，你要加入终结点的一个暴露信息的一个形式，这个*的话代表所有种类，如果说我只暴露健康的，我只暴露内存或者 thread 的或者 bean 的，你只指定一个用逗号分割，有多个用逗号分割。

2. Micrometer

1）Micrometer 是多维度指标收集器，语言中立的 API。

2）通过类路径和配置，可以支持多系统导出数据，集成框架！

3）Spring Boot 2 Actuator 中包含的指标收集工具。

4）Spring Boot 1.5,1.4 和 1.3 支持，额外依赖项。

5）Micrometer 为 Spring Boot 1 中 counters 和 gauges 增强功能。

6）例如， Micrometer Timer 能够生成吞吐量，总时间，最近样本的最大延迟，预先计算的百分位数，百分位柱状图相关的时间序列和 SLA 边界计数。

7）支持更多的监控工具如 Prometheus, Datadog, Wavefront, SignalFx, Influx, etc。

8）Spring Boot 2 强化特性。

Micromeat 叫微型指标收集器，它是帮你收集并发，吞吐量相关的那些信息 qps，Wps 每秒的读取数量每秒的写入数量，这些数据都可以收集起来以后做曲线图，Micromeat 的话叫基准测量。

Count 是个计数器，这个接口累计被调用次数，还有我们说高阶的话，有可能统计的我们说是不一样的，里面我们说的是针对不同的数据类型，它不同的统一指标，它的数量是不一样，比如我们统计的可能是过去一秒钟它的并发每秒钟的并发的分值，这都

不一样。

3. Spring Boot 2.0 增强 Micrometer

1）JVM 各种参数使用率

2）各种内存和缓冲池

3）与垃圾收集有关的统计

4）线程利用率

5）加载/卸载的类的数量

6）CPU 使用率

7）Spring MVC 和 WebFlux 请求延迟时间

8）RestTemplate 延迟时间

9）缓存使用率

10）数据源使用率，包括 HikariCP 池指标参数

11）RabbitMQ 连接工厂参数

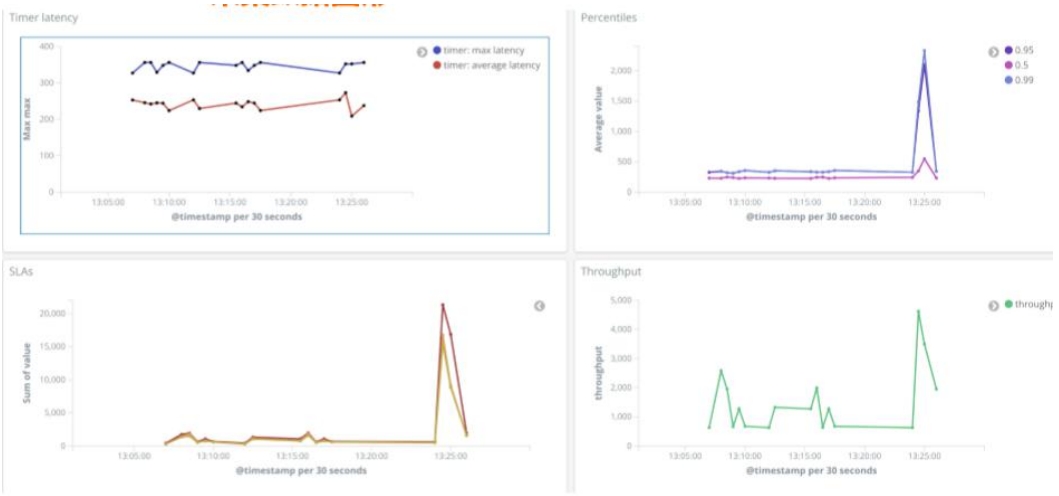
12）文件描述符使用率使

13）Logback：记录每个级别记录到 Logback 的事件数量

14）正常运行时间：报告正常运行时间表和表示应用程序绝对启动时间的固定计量表

15）Tomcat 使用情况

4. Micrometer 采集数据图形



5. Micrometer 支持的监控系统

1）Netflix Atlas

2）CloudWatch

3）Datadog

4）Ganglia

5）Graphite

6）InfluxDB

7）JMX

8）New Relic

9）Prometheus

10）SignalFx

11）StatsD (Etsy, dogstatsd, Telegraf, and proprietary formats)

12）Wavefront

13）AppOptics

14）Azure Application Insights

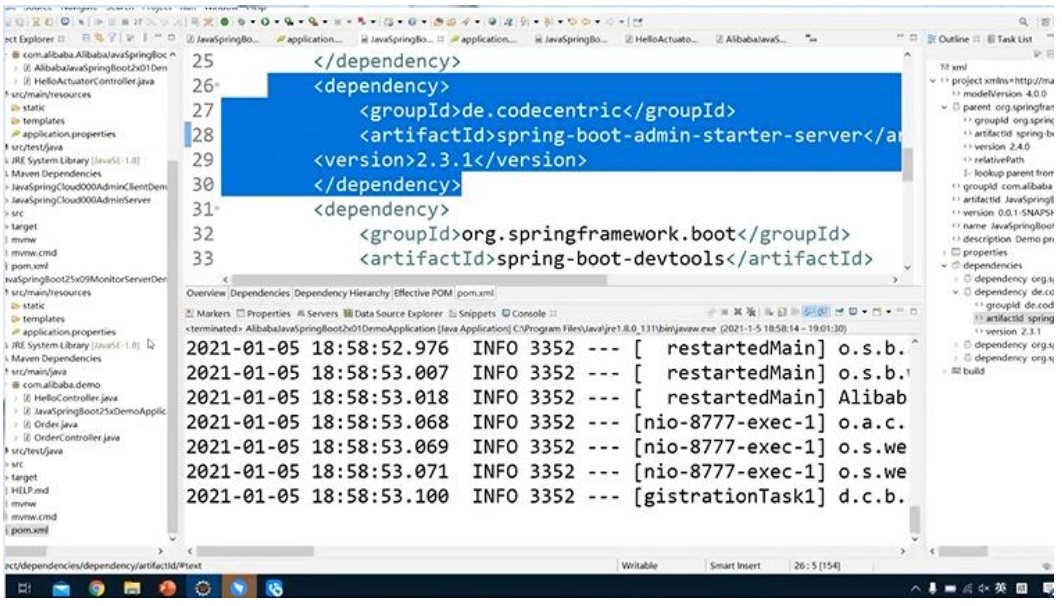
15）Dynatrace

16）Elasticsearch

17）StackDriver

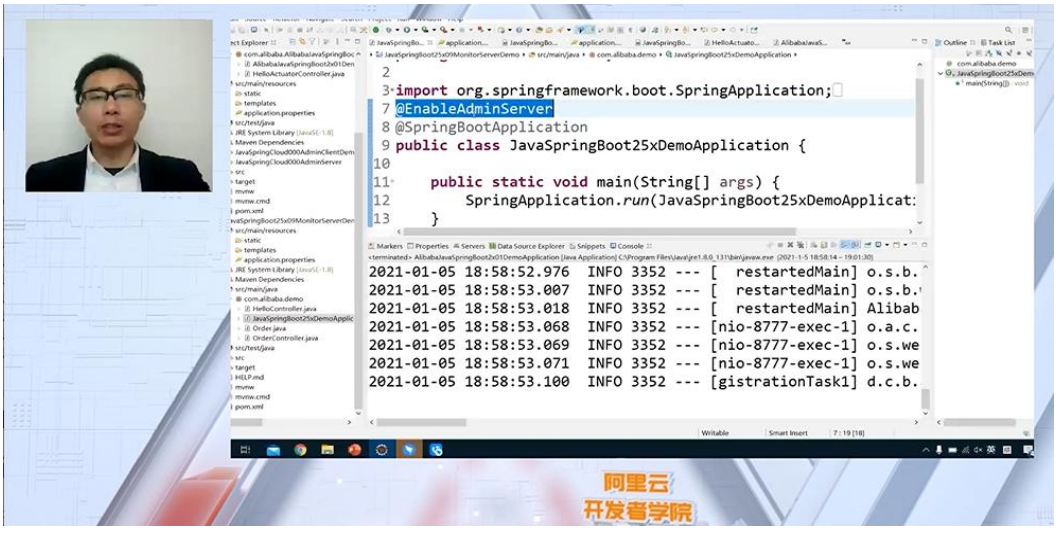
三、Java Spring Boot 2.5 监控实战

1. 开发监控服务端 Spring Boot Admin Server



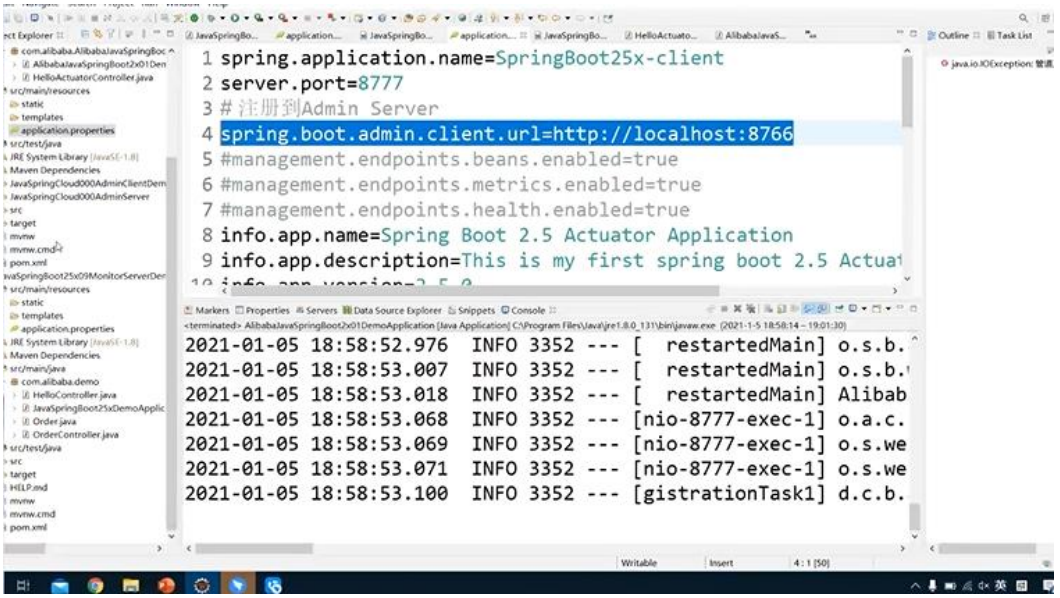
Spring boot 的域名的一个 server 端，收入端一定要有，而且要启用一个很重要的注解。

注解：

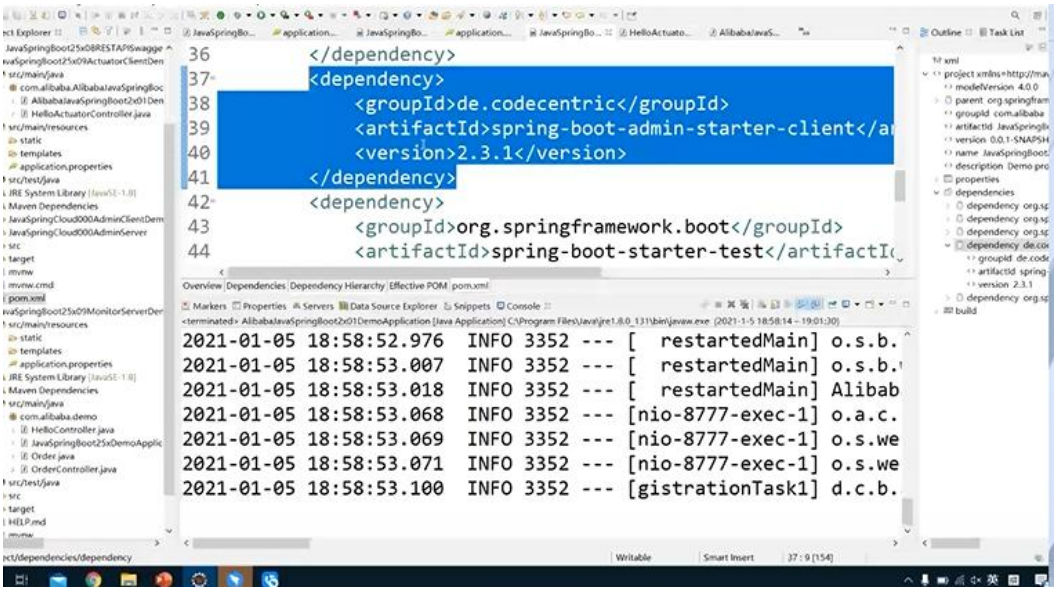


这里面模拟了一个被监控的思维，模特成语主要里面最主要的就是它配置文件里面一定要暴露监控的数据，如果暴露两种的话，在这里面直接来配置就可以了，否则是暴露所有，

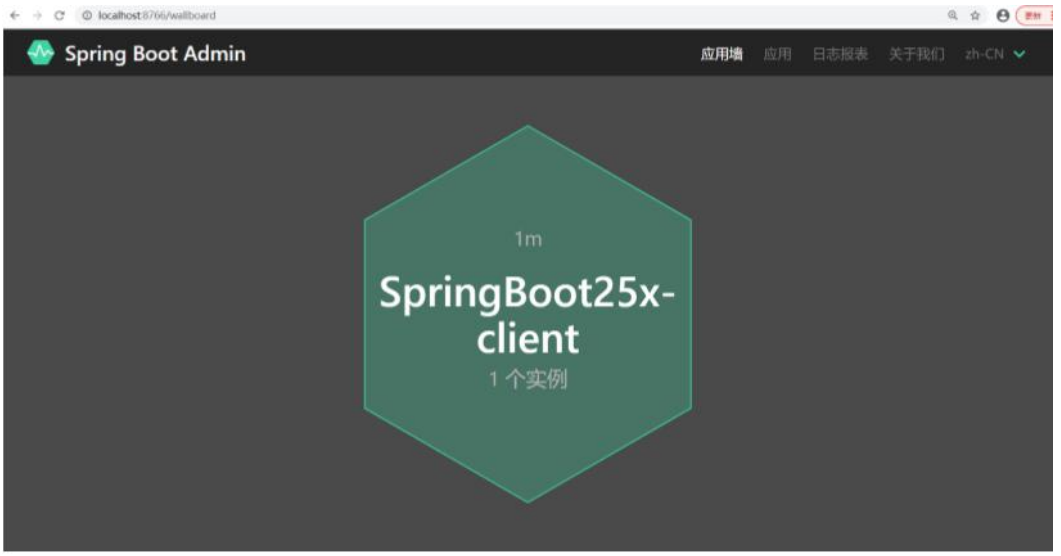
另外的话就是说这里面作为作为被监控方作为客户端程序的话，你这里面要配置一下监控中心 admin 的例子。



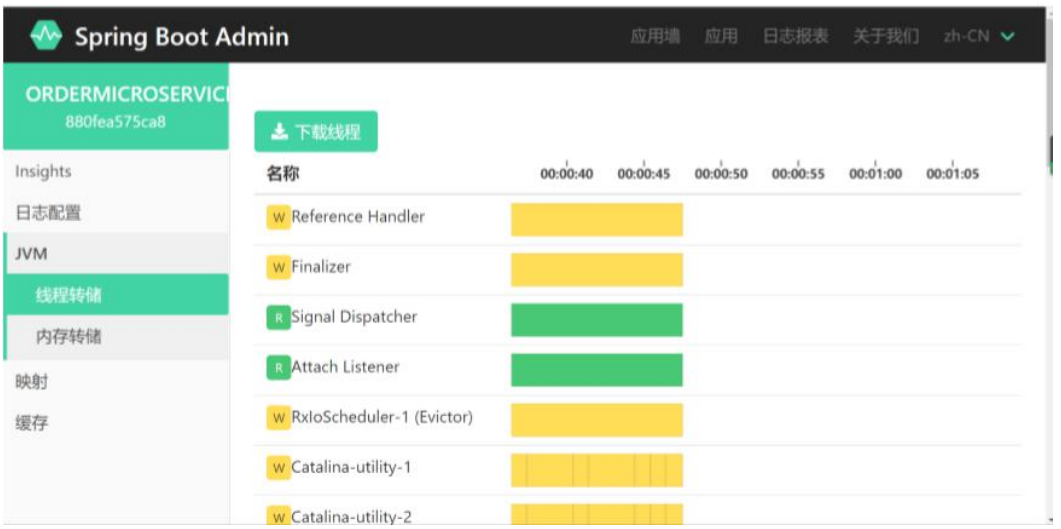
Admin 一个 spring 的客户端的依赖要加进来，客户端程序就具备了数据采集加数据上传的功能。



2. Spring Boot Admin 监控中心



3. Spring Boot Amin 监控中心



10. Spring Boot2.5 实战 Docker 容器

内容简介：

一、Spring Boot2.5.x 部署方式

二、Spring Boot 实战 Docker 容器 DockerHub

三、阿里云 Docker 镜像仓库

四、Docker 分布式集群架构

五、Docker 容器常用命令

六、Spring Boot 2.5 Docker 制作镜像

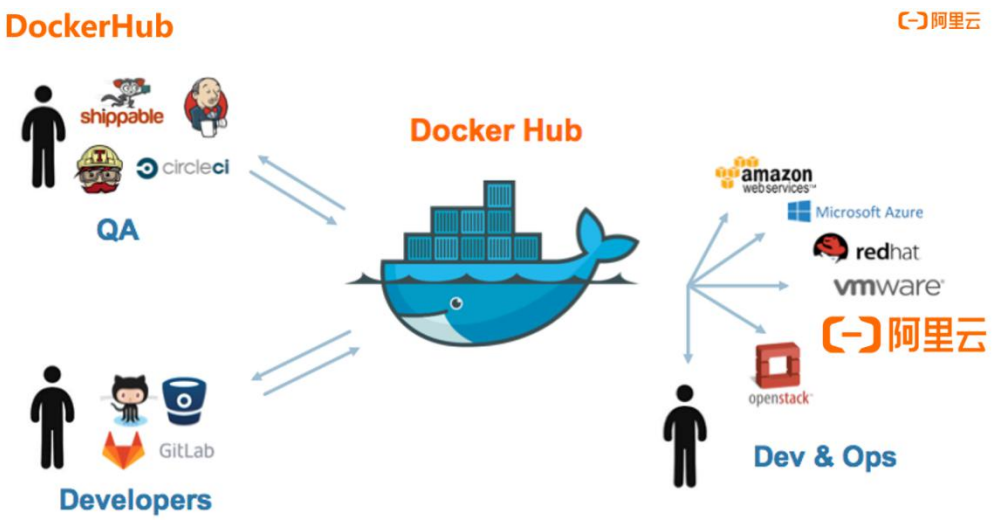
七、演示

八、高级面试题

一、Spring Boot2.5.x 部署方式

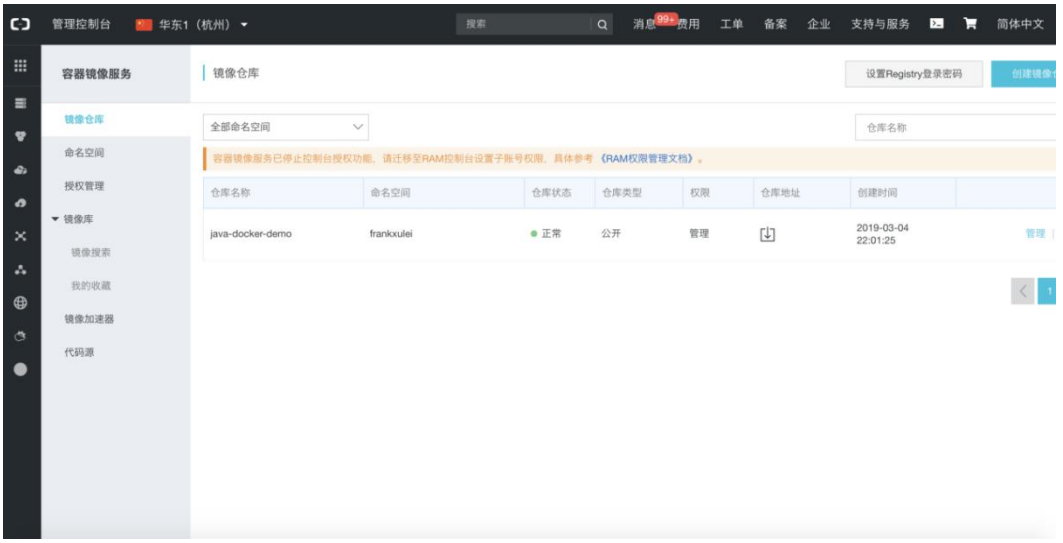
由于 Spring Boot 内嵌了很多 Web 容器，现在部署 Spring Boot 应用相对比较简单。这里指的 Web 应用可以用这个方法部署。如果要部署普通的应用，比如命令方式，也可以用 doctor 方式或者用其他的部署工具结合服务器或者远端的云端服务器进行自动化部署。

二、Spring Boot 实战 Docker 容器 DockerHub



Docker 作为一个容器工具，它实际是一个平台、一个生态，它包括 Docker 的服务器端、客户端以及仓库和可视化管理界面。

我给大家讲课的时候用的是 windows10，两年前 windows1 不支持 Docker，现在基本上就可以了，因为要开虚拟化技术。如果你用 Linux 环境或者 Mac OS，Docker 环境是比较容易安装。



DockerHub 本身是个类似于一个镜像的仓库，只不过这个服务器基本上都在美国。国内的话像阿里跟 Docker 签了合作协议，阿里云构建了一个 Docker 中国大陆的镜像服务，可以创建 Docker 的私有仓库。阿里云注册账号后在自己的账号里面可以看到 Docker 的一个镜像。Docker 应用程序的一个构建和发布标准化大规模集群的部署提供了非常便捷的操作方式。

Docker 本身也有自己的管理工具叫 Docker swarm。当有了 Docker 专属的 DockerHub 地址之后，再想上传到镜像的时候，可以上传到阿里云的 Docker 仓库里面。

在上传到阿里云仓库之前要先构建镜像，比如 MySQL、MongoDB、Redis 或者 Java 的一些标准主机镜像以外，也可以构建自己的镜像。

三、阿里云 Docker 镜像仓库

1）国内第一个提供 Docker 服务的云计算公司

2）2016 年云栖大会宣布与 Docker 官方合作

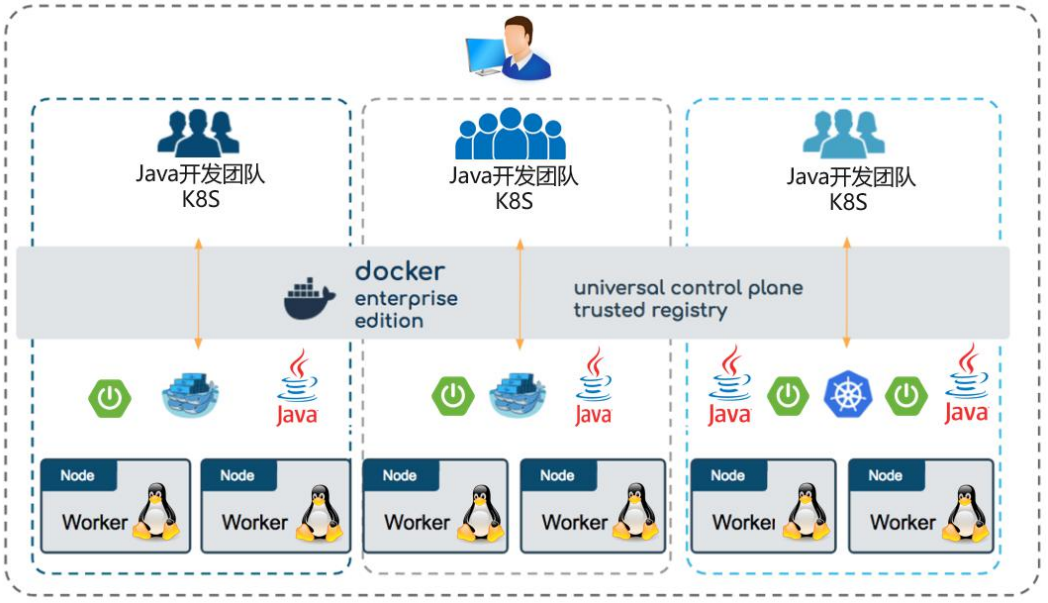
3）提供 Docker Images 镜像加速

4）提供景象仓库

5）免费注册使用

6）Docker Hub 中国站与 Docker Hub 完全一致

四、Docker 分布式集群架构



Docker 主要是方便开发和运维团队做大规模集群化部署。

五、Docker 容器常用命令

1）搜索镜像：sudo docker search java

2）拉取镜像：sudo docker pull java

3）运行镜像：docker run

4）获取帮助：docker help

5）构建镜像：docker build

6）提交镜像：docker commit

7）新建镜像：docker create

8）运行镜像：docker run

9）重启镜像：docker restart

10）查询全部：docker images

11）查看信息：docker info

12）推送镜像：docker push

1. DockerFile 命令

Docker 构建的早期需要 DockerFile，就是 Docker 构建了一个命令文件。Docker 基于这个文件构建镜像并且打包镜像。

1）Docker 镜像配置文件

2）脚本编写

3）脚本文件

4）一系列命令和参数构成的脚本

5）这些命令应用于基础镜像

6）并最终创建一个新的镜像

2. DockerFile 重要指令

1）FROM 指定基础镜像文件

2）MAINTAINER authors_name 作者

3）RUN 运行特殊命令，比如下载 JDK

4）SER 命令用于设置运行容器的 UID

5）VOLUME 指定容器访问目录

6）WORKDIR 运行目录

7）ENV 环境变量，如 ENV LANG en_US.UTF-8

8）CMD 容器执行的命令 CMD "echo" "Hello docker!"

9）ADD 复制文件到目标文件夹

10）COPY 复制，类似 ADD

11）EXPOSE 暴露端口

12）ENTRYPOINT 入口，命令，只有一个不能被 Run 覆盖

六、Spring Boot 2.5 Docker 制作镜像

1. 环境需求

- docker (1.6.0 or above)
- jdk 1.8
- Maven 3.0+或者 Gradle 2.3+

2. Spring Boot 2.4 全新 docker 构建工具

- Spring Boot 2.4 推出了自己的 docker 构建工具
- 一键构建 Docker 镜像，无需 Dockerfile
- 之前工具 spotify 、fabric8，配置插件使用，需要 Dockerfile
- jib-maven-plugin 是 Google18 年 7 月发布的 Java 镜像工具(支持 Maven 和 Gradle)，也无需 Dockerfile

- 整合在原有的 spring-boot-maven-plugin 中，
- 只需要配置对应目标仓库和主机信息即可完成镜像构建。
- 新命令：mvn spring-boot:build-image

七、演示

1. Dockerfile 制作镜像

- FROM java:8
- VOLUME /tmp
- ADD java-spring-boot-docker-0.1.0.jar app.jar
- RUN bash -c 'touch /app.jar'
- ENTRYPOINT

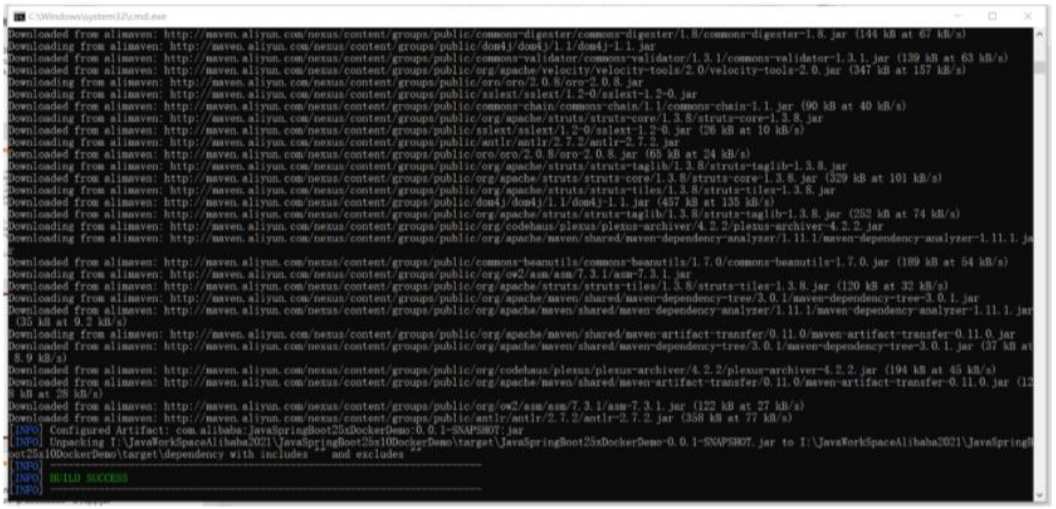
["java","Djava.security.egd=file:/dev/./urandom","jar","/app.jar"]

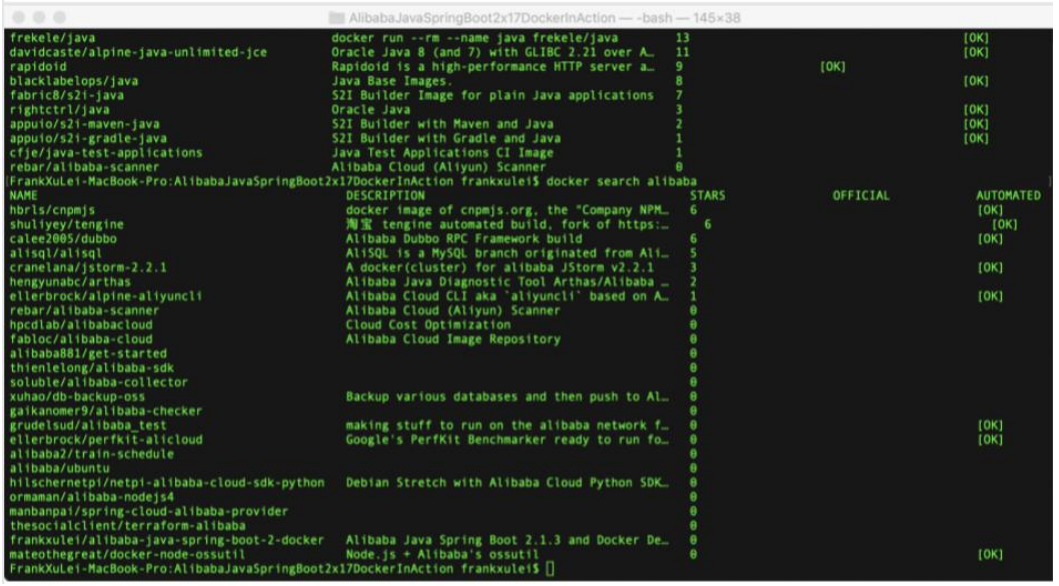
2. Build Docker Image with Maven

- <plugin>
- <groupId>com.spotify</groupId>
- <artifactId>docker-maven-plugin</artifactId>
- <version>0.2.3</version>
- <configuration>
- <imageName>${docker.image.prefix}/${project.artifactId}</imageName>
- <dockerDirectory>src/main/docker</dockerDirectory>

- <resources> • <resource> • <targetPath>/</targetPath>
- <directory>${project.build.directory}</directory>
- <include>${project.build.finalName}.jar</include>
- </resource>
- </resources>
- </configuration>
- </plugin>

3. Maven 打包





八、高级面试题

1. Docker 是什么？解决什么问题？
2. Docker 的优势？
3. Swarm 工具和 Docker 的关系
4. K8s 优势是什么？
5. 如何安装 Docker 容器？
6. 如何制作、推送 Docker 镜像？
7. 如何搜索、拉取、启动 Docker 镜像？
8. Docker 如何容器部署 Spring Boot 2.5.x 和微服务

扫一扫

免费领取同步课程

钉钉扫一扫

进入官方答疑群

开发者学院【Alibaba Java 技术图谱】

更多好课免费学

阿里云开发者“藏经阁”

海量电子书免费下载