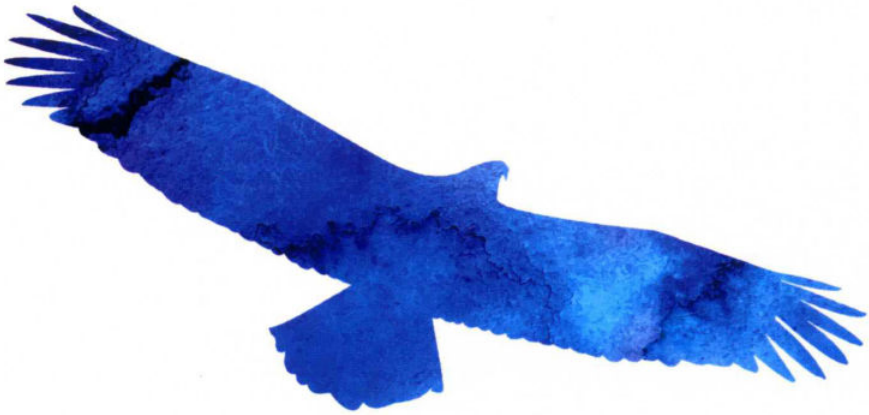
140个实战案例，完美演示微服务的方方面面

丰富的架构图示+手把手步骤学习，轻松掌握微架构设计与开发

名师讲坛



Java 微服务架构实战

SpringBoot +SpringCloud +Docker +RabbitMQ

李兴华 ◎ 编著

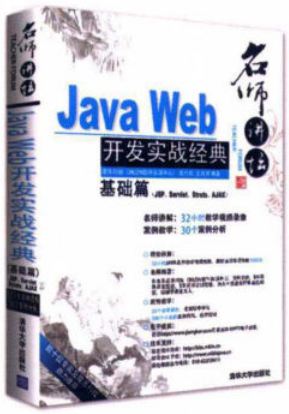
微服务架构技术宝典

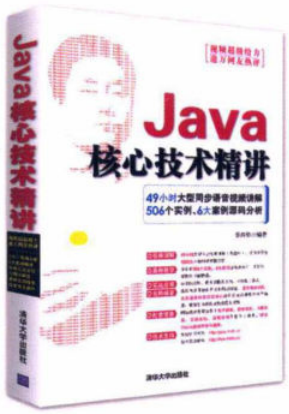
Java名师15年开发/教学经验集萃

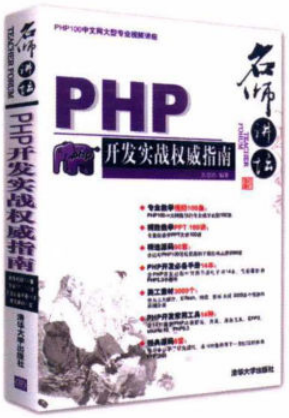
源代码下载

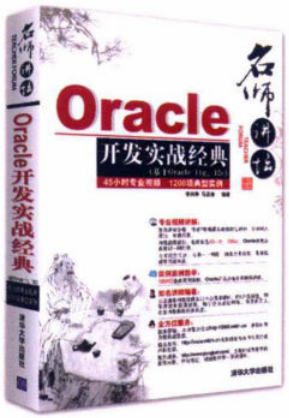
清華大學出版社

技术与经验荟萃　成就光荣与梦想









文泉云盘
获取学习资源

清华社官方微信号
扫我有惊喜

上架建议：软件・编程
ISBN 978-7-302-50607-2
9 787302 506072 >
定价：69.80元

名师讲坛——Java 微服务架构实战

（SpringBoot+SpringCloud+Docker+RabbitMQ）

李兴华 编著

清華大學出版社

北 京

内 容 简 介

Java 微服务架构是当下最为流行的软件架构设计方案，可以快速地进行代码编写与开发，维护起来也非常方便。利用微架构技术，可以轻松地实现高可用、分布式、高性能的项目结构开发，同时也更加安全。

本书一共 15 章，核心内容为 SpringBoot、SpringCloud、Docker、RabbitMQ 消息组件。其中，SpringBoot 是 SpringMVC 技术的延伸，使用它进行程序开发会更简单，服务整合也会更容易。SpringCloud 是当前微架构的核心技术方案，属于 SpringBoot 的技术延伸，它可以整合云服务，基于 RabbitMQ 和 GITHUB 进行微服务管理。除此以外，本书还重点分析了 OAuth 统一认证服务的应用。

本书适用于从事 Java 开发且有架构与项目重构需求的读者，也适用于相关技术爱好者，同时也可作为应用型高等院校及培训机构的学习教材。

图书在版编目（CIP）数据

Java 微服务架构实战：SpringBoot+SpringCloud+Docker+RabbitMQ/李兴华编著. —北京：清华大学出版社，2020.1
（名师讲坛）
ISBN 978-7-302-50607-2

I. ①J… II. ①李… III. ①JAVA 语言-程序设计 IV. ①TP312.8

中国版本图书馆 CIP 数据核字（2018）第 151303 号

责任编辑： 贾小红
封面设计： 魏润滋
版式设计： 楠竹文化
责任校对： 马军令
责任印制： 李红英

出版发行： 清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者： 北京密云胶印厂
经　　销： 全国新华书店
开　　本： 185mm×260mm　　**印　　张：** 19　　**字　　数：** 475 千字
版　　次： 2020 年 1 月第 1 版　　**印　　次：** 2020 年 1 月第 1 次印刷
定　　价： 69.80 元

产品编号：080068-01

前　言

我们在用心做事，做最好的教育，写最好的原创图书。

笔者是一名从事 Java 开发快二十年的技术爱好者，一位普通的培训班老师，喜欢和学生们一边开着玩笑，一边教会他们当下流行与实用的技术。很多时候我会跟学生说："信息产业是一个不断发展变化的行业，没有人可以精确预测这个行业的未来发展方向，更没有人可以在这个行业里拥有绝对的技术实力。同样，也没有永远不过时的技术。我们能做的只是努力地学习与提升，每一天都要在踩坑与填坑的路上不断爬行，磕磕碰碰习惯了，解决问题所花费的时间就越来越少了。想要在这个行业走得长远，一定要喜欢这个行业，喜欢钻研。"

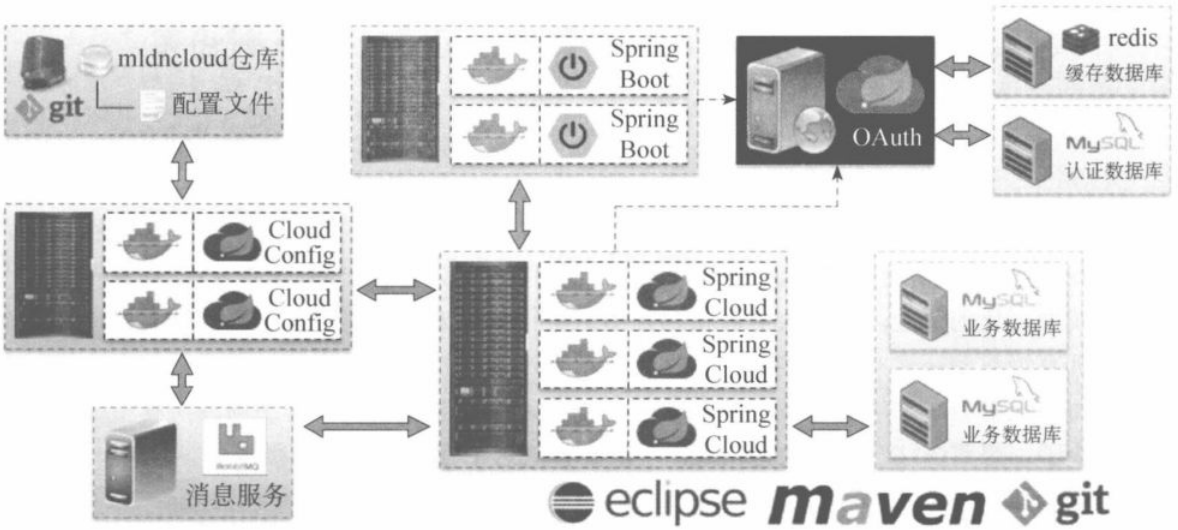
遥想起 2003 年开源风在中国兴起时，SSH（Spring 1.x + Struts 1.x + Hibernate 2.x）整合开发框架是当时最大的技术亮点。作为开发者的我们，最大的感受是再也不需要去编写那些重复的代码了，利用开发框架我们几乎可以解决当时所有的问题。然而技术的经典是短暂的，随着时间的流逝，SSH 的光环也不再辉煌。后来，有了 SSH2（Spring 2.x + Struts 2.x + Hibernate 3.x），又有了 SSM（Spring + Shiro + MyBatis）。随着开发框架的不断增加，以及 Spring 对各类开发框架的不断支持，新的问题出现了——参与整合的配置文件过多，项目的集成化太高。大家转而开始寻找新的解决方案。就在所有人都认为 Pivotal 公司（Spring 项目所属公司）已经停滞不前的时候，其在 2016 年推出了一套完善的轻量级分布式解决方案，就是今天流行的微架构（或称微服务），之中的主要技术手段是 SpringBoot + SpringCloud。

微架构的出现，很好地适应了这个时代对快速发展变化的要求。它不再提倡一体化的项目设计，而是对项目进行有效的"业务区"（可以简单理解为不同的子系统）划分，并利用合理的技术对业务性能做出提升和改善，同时又极大地简化了配置文件的使用与 profile 配置。总而言之，微架构是开发之中看起来非常简单的一种实现技术，但简单的背后考究的却是开发者对于开源技术的熟练程度。

SpringBoot 作为一种 Web 整合开发框架，很好地解决了 Web 程序的编写困难，可以更简单、高效地实现 MVC 设计模式。更为重要的是，它可以轻松地整合当前各类主流的开发项目，如消息组件、SQL 数据库、NoSQL 数据库、邮件服务等，因此能极大地缩短项目的开发周期，更快地响应客户的需求变更。SpringCloud 作为 SpringBoot 的延续，可以基于 Restful 流行架构实现 RPC 业务中心的搭建，可以基于消息组件实现远程配置动态的抓取，还可以与 Docker 相结合，采用虚拟化手段实现便捷的云服务管理。可以说，微架构的出现与云时代是密不可分的。

本书是笔者多年开发经验的总结，写作时力求能一针见血地分析透 Java 微服务的设计架构（见下图）与各类技术实现。全书围绕着当前的主流方案（高性能+高可用+分布式）进行展开，

不仅讲解了所有微架构中的内容，还给出了真实有效的学习案例；不仅可以与虚拟化 Docker 整合开发，还可以实现大型企业分布式授权 OAuth 解决方案。可以说，本书就像 Java 微服务实现架构的一个技术宝典，读者学习后完全可以直接在实际项目之中进行应用。另外，由于微架构涉及到的技术非常广泛，对于某些技术还不十分清楚的读者，可以登录魔乐科技网站（www.mldn.cn）进行视频学习。



笔者崇尚原创，所出版的图书也均为原创。笔者将“技术实现优先”这一原则贯穿于全书，采用步骤分解的模式详细讲解每一步的开发，希望读者可以通过本书学习到微服务的技术精髓。另外，由于技术更新迭代过快，加之本人水平有限，书中难免有表达不到位或不明确的地方，欢迎读者批评指正，万分感谢。

创作不易。感谢我最爱的妻子和我的家人，是你们的付出与支持才让我可以安心创作，同时也祝福我年幼的儿子可以健康快乐地成长。

本书特色

- ☑ 资深 Java 讲师进行技术剖析，全面把握学习命脉，问题分析一针见血。
- ☑ 140 个课程案例，完美演示微服务的方方面面。
- ☑ 基于 Maven 实现项目管理，与真实项目完美衔接。
- ☑ 丰富的架构图示说明，轻松掌握微架构设计方案。
- ☑ 手把手步骤学习法，轻松掌握微架构开发。
- ☑ OAuth 使用分析与代码实现，掌握企业级 RPC 认证与授权解决方案。
- ☑ 微服务与 Docker 虚拟化技术结合使用，轻松实现云服务。

本书章节安排

全书涉及到的技术包括：SpringBoot、Thymeleaf、Jetty、Redis 整合、C3P0 整合、Druid 整合、MyBatis 整合、ActiveMQ 整合、RabbitMQ 整合、Kafka 整合、Shiro 整合、SpringDataJPA

整合、Mail 整合、Actuator 监控、Restful、RestTemplate、Eureka、Ribbon、Feign、Hystrix、Turbine、Zuul、SpringCloudConfig、SpringCloudBus、SpringCloudStream、SpringCloudSleuth、Zipkin、OAuth、RabbitMQ 和 Docker。

考虑到学习层次，本书共分为 3 个组成部分：SpringBoot 篇、SpringCloud 篇和微服务辅助篇。

第一部分：**SpringBoot 篇**

- ☑ **第 1 章　SpringBoot 编程起步**：本章将为读者讲解 SpringBoot 的发展背景与 SpringBoot 编程起步。
- ☑ **第 2 章　SpringBoot 程序开发**：本章将为读者详细讲解 SpringBoot 开发常用的各项技术，包括代码测试、Jetty 配置、资源加载、访问路径、profile 配置、项目打包等。
- ☑ **第 3 章　Thymeleaf 模板渲染**：Thymeleaf 是模板技术，也是当下 Web 开发中使用最多的一项技术，在 SpringBoot 中默认支持有此模板使用，本章将为读者讲解 Thyemeleaf 之中的使用语法以及与 JSP 语法的关联。
- ☑ **第 4 章　SpringBoot 与 Web 应用**：主要讲解 https 协议整合、Tomcat 发布、全局异常处理、文件上传等。
- ☑ **第 5 章　SpringBoot 服务整合**：主要讲解 C3P0、Druid、MyBatis、SpringDataJPA、ActiveMQ、RabbitMQ、Kafka、Redis、Shiro、Mail、Actuator 监控等组件的整合应用。

第二部分：**SpringCloud 篇**

- ☑ **第 6 章　SpringCloud 简介**：主要讲解 RPC 技术的主要作用及 SpringCloud 技术实现架构。
- ☑ **第 7 章　SpringCloud 与 Restful**：主要讲解 Restful 架构的基础实现方案、RestTemplate 调用微服务以及 SpringSecurity 基础认证处理。
- ☑ **第 8 章　Eureka 注册服务**：主要讲解 Eureka 的作用、Eureka 微服务创建、Eureka 集群搭建、打包部署等。
- ☑ **第 9 章　SpringCloud 服务组件**：主要讲解 Ribbon 负载均衡、Feign 接口转换、Hystrix 熔断机制以及 Zuul 代理机制。
- ☑ **第 10 章　SpringCloudConfig**：与 GitHub 结合实现分布式配置文件管理、加密处理、SpringCloudBus 更新服务。
- ☑ **第 11 章　SpringCloudStream**：讲解微服务中信息采集的搭建，主要与 RabbitMQ 整合。
- ☑ **第 12 章　SpringCloudSleuth**：讲解微服务调用监控跟踪、Zipkin、数据采集。
- ☑ **第 13 章　OAuth 认证管理**：分析 SpringSecurity 实现方案缺陷、OAuth 与 RPC 结合流程，并基于 SQL 数据库与 Redis 数据库实现 OAuth 认证与授权管理。

第三部分：**微服务辅助篇**

- ☑ **第 14 章　RabbitMQ 消息组件**：RabbitMQ 与 Spring 微服务有着密不可分的关联，本章将为读者讲解 RabbitMQ 的安装、管理、Java 开发与集群使用。

☑ **第 15 章　Docker 虚拟化容器：**虚拟化与云开发是流行话题，本章主要讲解 Docker 虚拟化容器管理技术，同时讲解微服务与 Docker 的整合开发以及 DockerCompose 组件的使用。

寄语读者

本书全篇由笔者根据实践项目与教学经验总结而来，虽经过再三斟酌和审校，仍难免存在技术理解上的偏差和解释不到位的地方，欢迎读者批评指正。您的宝贵建议将帮助我们修正此书，大家一起努力，将传道、授业、解惑贯彻到底。

本书用到的程序源代码，读者可扫描图书封底的“文泉云盘”二维码获取其下载方式，也可登录清华大学出版社网站（www.tup.com.cn）进行下载。技术学习部分，读者可登录魔乐科技官网（http://www.mldn.cn）及沐言优拓官网（http://www.yootk.com）进行学习，也可登录笔者的新浪微博进行留言交流。

最后，希望本书成为您的良师益友。祝您读书快乐！

李兴华

目 录

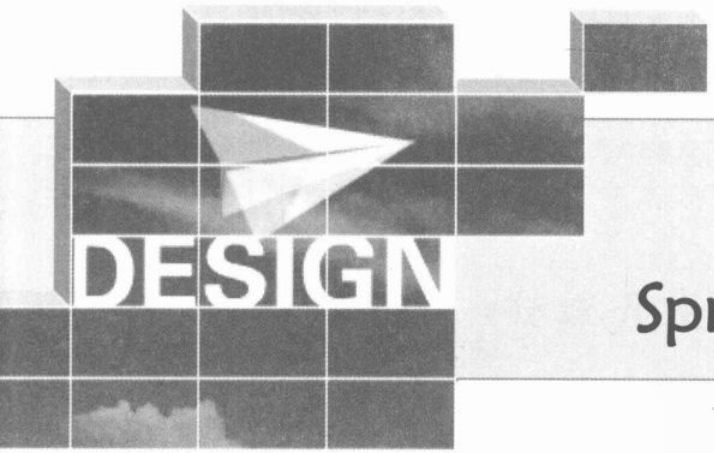
Contents

第一部分 SpringBoot 篇

第二部分 SpringCloud 篇

第三部分　微服务辅助篇

第一部分



SpringBoot 篇

- SpringBoot 与 Restful 标准
- SpringBoot 微服务创建
- Thymeleaf 语法标准
- SpringBoot 与服务整合

第1章

SpringBoot 编程起步

通过本章学习，可以达到以下目标：

1. 理解基于 Maven 的传统项目开发问题。
2. 理解 SpringBoot 开发框架的主要作用。
3. 编写第一个 SpringBoot 程序。

SpringBoot 是当下最为流行的 Java Web 端开发框架，该框架由 Spring（Pivotal 公司）开源组织提供，使用该框架可以解决传统项目之中混乱的 Maven 依赖管理问题，同时可以基于 Maven 快速进行项目的打包与发布。

1.1 传统开发中痛的领悟

在 Java 项目开发中，MVC 已经成为了一种深入人心的设计模式，几乎所有正规的项目之中都会使用到 MVC 设计模式。采用 MVC 设计模式可以有效地实现显示层、控制层、业务层、数据层的结构分离，如图 1-1 所示。

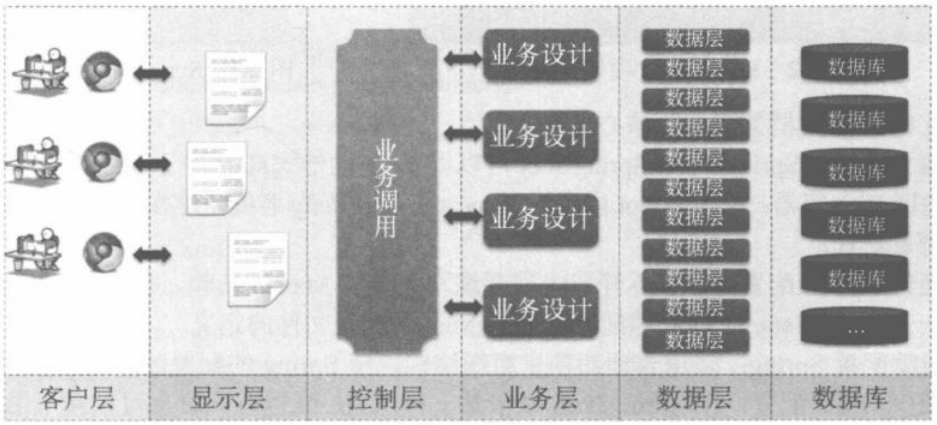


图 1-1 MVC 设计模式

虽然 MVC 开发具有良好的可扩展性，但是在实际的开发过程中，许多开发者依然会感受到如下的问题。

☑ 采用原生 Java 程序实现 MVC 设计模式时，一旦整体项目设计不到位，就会存在大量的重复代码，并且项目维护困难。

- ☑ 为了简化 MVC 各个层的开发，可以引用大量的第三方开发框架，如 Spring、Hibernate、MyBatis、Shiro、JPA、SpringSecurity 等，但这些框架都需要在 Spring 中实现整合，其结果就是会存在大量的配置文件。
- ☑ 当使用一些第三方的服务组件（如 RabbitMQ、Kafka、JavaMail 等）时，需要编写大量重复的配置文件，而且还需要根据环境定义不同的 profile（如 dev、beta、product）。
- ☑ 使用 Maven 作为构建工具时，需要配置大量的依赖关系，且程序需要被打包为*.war 文件并部署到应用服务器上才可以执行。
- ☑ Restful 作为接口技术应用得越来越广泛，但如果使用 Spring 来搭建 Restful 服务，则需要引入大量的 Maven 依赖库，并且需要编写许多的配置文件。

基于以上种种因素，很多人开始寻求更加简便的开发方式，而遗憾的是，这种简便的开发没有被官方的 JavaEE 所支持。JavaEE 官方支持的技术标准依然只提供最原始的技术支持。

1.2 SpringBoot 简介

SpringBoot 是 Spring 开发框架提供的一种扩展支持，其主要目的是希望通过简单的配置实现开发框架的整合，使开发者的注意力可以完全放在程序业务功能的实现上，其核心在于通过“零配置”的方式来实现快速且简单的开发。图 1-2 显示了 Spring 官方网站中 SpringBoot 项目，图 1-3 显示了 SpringBoot 当前的开发版本。

图 1-2 SpringBoot 项目站点

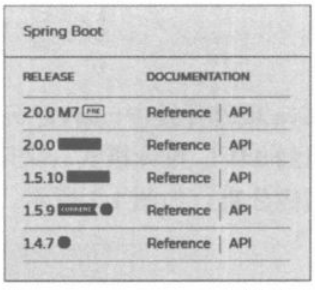


图 1-3 SpringBoot 支持版本

Spring Boot 开发框架有如下核心功能。

- ☑ 独立运行的 Spring 项目：SpringBoot 可以以 jar 包的形式直接运行在拥有 JDK 的主机上。
- ☑ 内嵌 Web 容器：SpringBoot 内嵌了 Tomcat 容器与 Jetty 容器，这样可以不局限于 war 包的部署形式。
- ☑ 简化 Maven 配置：在实际开发中需要编写大量的 Maven 依赖，在 SpringBoot 中会提供一系列使用 starter 的依赖配置来简化 Maven 配置文件的定义。
- ☑ 自动配置 Spring：采用合理的项目组织结构，使 Spring 的配置注解自动生效。
- ☑ 减少 XML 配置：在 SpringBoot 中依然支持 XML 配置，同时也可以利用 Bean 和自动配置机制减少 XML 配置文件的定义。

1.3 SpringBoot 编程起步

SpringBoot 编程需要依赖于 Maven 或 Gradle 构建工具完成，这里将直接使用 Maven 进行开

发，同时利用 Eclipse 来建立 Maven 项目。

1. 在 Eclipse 中创建一个新的 Maven 项目，项目类型为 quickstart，如图 1-4 所示。

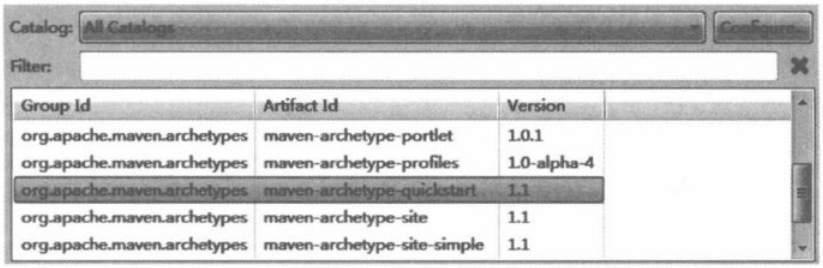


图 1-4　Eclipse 创建 Maven 项目

2. 设置 Maven 项目的信息（Group Id、Artifact Id、Version 等），本例建立的项目名称为 bootstart，如图 1-5 所示。

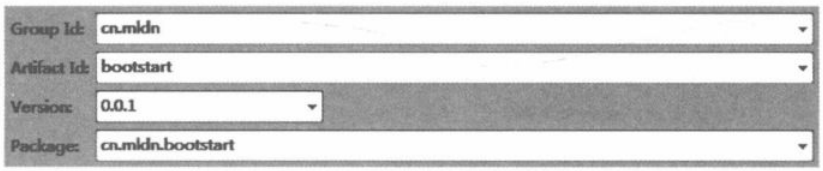


图 1-5　设置 Maven 的配置信息

3. 修改 pom.xml 配置文件，追加 SpringBoot 的依赖配置与相关插件。

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>                                          <!-- 引入SpringBoot支持 -->
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.5.9.RELEASE</version>
    </parent>
    <groupId>cn.mldn</groupId>
    <artifactId>bootstart</artifactId>
    <version>0.0.1</version>
    <packaging>jar</packaging>
    <name>bootstart</name>
    <url>http://maven.apache.org</url>
    <properties>
        <jdk.version>1.8</jdk.version>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>
    <dependencies>
```