构建基于 Dubbo 框架的 Spring Boot 微服务*

赵子晨[1]　朱志祥[1]　蒋来好[2]
（1. 西安邮电大学　西安　710061）（2. 陕西省信息化工程研究院　西安　710061）

摘　要　随着微服务理念逐渐深入人心和相关技术的不断成熟，越来越多的企业选择用微服务架构来构建自己的应用系统，这很好地克服了应用系统在后期版本升级、系统扩展等传统软件生产模式中存在的问题，实现了高效可重复的标准化微服务生产的目标。论文提出了一种 Dubbo+Spring Boot 构建微服务的方法，利用 Dubbo 高性能和透明化的远程服务调用，以 Spring Boot 的形式标准化开发，并且运用 Maven 来自动化管理项目。实验测试表明，开发基于 Dubbo 框架的 Spring Boot 微服务具有简单、高效、标准化的特点，为海量微服务的开发和运维提供重要的参考价值。

关键词　微服务；Dubbo；Spring Boot
中图分类号　TP311.5　　**DOI**：10. 3969/j. issn. 1672-9722. 2018. 12. 030

Build Spring Boot Microservice Based on Dubbo Framework

ZHAO Zichen[1]　ZHU Zhixiang[1]　JIANG Laihao[2]
（1. Xi'an University of Posts and Telecommunications，Xi'an　710061）（2. Institute of Communication Technology，Xi'an　710061）

Abstract　With the micro-service concept gradually becoming popular and relevant technology continuing to be mature，more and more enterprises choose to use micro-service framework to build their own application system. This is a good way to overcome the problems existing in the traditional software production model in the later version upgrade and system expansion of the application system，and achieves a highly efficient and repeatable standardization of micro-service production. This paper proposes a method to build the micro-service by Dubbo + Spring Boot，taking advantage of Dubbo's high-performance，transparent remote service invocation and standardized development of Spring Boot，and using Maven to automate management projects. Experimental tests show that the Spring Boot micro-service based on Dubbo framework is simple，efficient and standardized，which provides important reference value for the development，operation and maintenance of the mass micro-service.

Key Words　microservice，Dubbo，Spring Boot
Class Number　TP311.5

1　引言

近几年来，“微服务架构”的概念已经在软件开发领域逐渐获得认可和实践。微服务作为“面向服务架构”（SOA）的下一代继承者，其实际上也可以被归为“分布式系统”这一范畴，并且进一步发扬面向服务架构中的许多观点和实践。但是，它们区别的地方在于每个单独的服务所应承担的功能范围。在 SOA 中，每个服务将负责处理规模更广的功能与数据领域，而微服务自从问世以来，业界普遍认为，它所负责的部分是管理一个单独的数据领域，以及围绕着该领域的相关功能。使用分布式系统这种方式的目的在于将整体性的服务基础设施解耦为独立的可扩展的子系统，可以通过垂直分片的方式将这些子系统依赖合并在一起，并通过一种统一的传输方式将它们相互连接[1]。

微服务架构对于服务之间交互的状态性这方面有着严格的要求，不管采用了哪一种底层协议，微服务都需要保证通信的无状态性，并且遵守 RESTful 模式风格来实现这一点，这在软件开发领域基本已经达成了一致的认可。Dubbo 就是一个完美的构建微服务的分布式服务框架，既可以实现 RPC 远程服务调用，又支持 REST 风格远程调用（HTTP + JSON/XML）。

* **收稿日期**：2018 年 6 月 10 日，**修回日期**：2018 年 7 月 25 日
作者简介：赵子晨，男，硕士研究生，研究方向：大数据处理与高性能计算。朱志祥，男，博士，教授，研究方向：信息安全研究。蒋来好，男，硕士研究生，研究方向：大数据处理与高性能计算。

使用Spring Boot开发基于Dubbo框架的微服务的原因就是,将Dubbo微服务以Spring Boot形式标准化了。如此一来既可以享受Spring Boot框架和周边的一系列研发支持,还可以用统一的形式发布、部署和运维。即Spring Boot可使软件开发者面对很多个相同操作接口的微服务,而不再是不同操作接口的微服务。

同时使用Maven+Docker进行项目的快速构建和可移植性部署,Maven可以使项目管理更加轻松,Docker容器作为一种轻量级的虚拟化技术,提供和虚拟机相同的功能但又更加的强大,简单易用,轻快便捷[2]。

2 微服务

从本质上来讲,微服务(Microservice)在很早之前就是已经存在的事物了,只是人们没有给它定义一个比较专业的名称而已。自SOA(Service Oriented Architecture)面向服务体系架构诞生以来,各大企业公司遵循SOA软件架构的思路,在服务化治理的道路上走了很长的时间,总结了服务化思路的一种最佳实践方向,这就是微服务。

微服务这个名字是和以前的服务化思路与实践相对比得来的。很多年前的服务实现和实施思路是把许多从开发到交付的功能给打包成一个特别大的服务单元,也即是单体式应用(一般会称之为Monolith),但是微服务的实现和实施思路是更强调功能趋向单一化的,即服务单元的小型化和微型化。如图1是Monolith Service和Microservice,齿轮表示服务的功能。

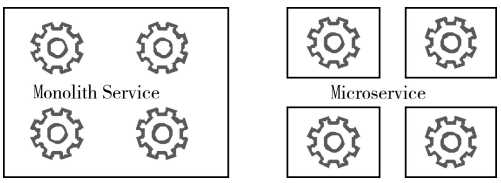

图1 Monolith Service(左)和Microservice(右)

所以从思路与理念上来说,微服务就是提倡尽量将复杂一点的功能进行拆分,把服务粒度做更小,让它能够独立承担对外服务的职责,顺着这个思路开发和交付的软件服务就称为“微服务”。

3 Dubbo

提到服务化和框架,国内大部分研发者第一印象或许会直接想到Dubbo这样的服务框架。Dubbo是阿里巴巴公司开源出来的SOA服务化治理方案的核心框架,它是一个分布式的专注于提供高性能和透明化的RPC远程服务调用,以及SOA服务治理的方案。其核心部分包含:

1)远程通讯:提供对多种基于长连接的NIO框架抽象封装,包括几种信息交换方式如,多种线程模型,序列化,以及“请求-响应”模式。

2)集群容错:提供基于接口方法的透明远程过程调用,支持多协议,以及软负载均衡,失败容错,地址路由,动态配置等。

3)自动发现:基于注册中心的目录服务,主要提供了服务消费方能动态的查找服务提供方,地址透明,服务提供方可以平滑增加或减少机器[5]等几项功能。

Dubbo的架构如图2所示。

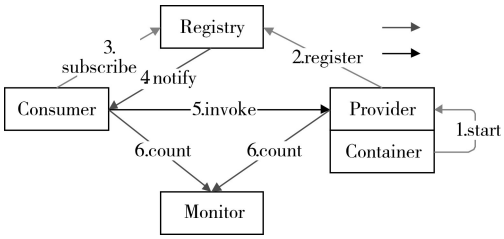

图2 Duboo架构

节点角色说明如下:

1)Provider:暴漏服务的服务提供方。

2)Consumer:调用远程服务的服务消费方。

3)Register:服务注册与发现的注册中心。

4)Monitor:监控中心负责统计服务的调取次数和调取时间。

5)Container:服务运行容器。

架构中各节点的调用关系说明如下(与图2中序号对应):

1)底层容器主要负责开启,加载,运行服务提供者。

2)服务提供者在容器启动时,向注册中心注册自己所提供的服务。

3)服务消费者在应用服务器启动时,向注册中心订阅自己所需的服务。

4)注册中心返回服务提供者地址列表给消费者,如果有变更,注册中心将基于长连接推送变更数据给消费者。

5)在注册中心的服务提供者列表中,服务消费者基于软负载均衡算法选择一个服务提供者进行调用,如果调取失败,注册中心经过负载均衡算法动态的再次分配给服务消费者另一台。

6)监控中心每分钟接受一次来自内存中的统计数据,来记录服务提供者和消费者的调用次数和调用时间。

基于以上 Dubbo 这些核心功能，对于开发功能趋向单一，数量众多，相互调用的 Spring Boot 微服务具有重大意义。Dubbo 可以提供以下功能：

1）像调用本地方法一样透明化的调用远程方法，只需要在配置文件中简单注入，没有任何 API 接口侵入。

2）软负载均衡及容错机制，可在内网替代 F5 等硬件负载均衡器，降低成本，减少单点。

3）Provider 提供的服务动态注册和发现，不再须要写死服务地址，注册管理中心基于 API 名查询服务提供者的 IP 地址，并且能够平滑添加或删除服务提供者[8]。

4　Spring Boot

随着微服务理念的盛行，一个流行的概念也随之诞生－微框架（Micro Framework），Spring Boot 是一个可使用 Java 构建微服务的微框架。Spring Boot 是 Spring 框架对“约定优先于配置（Convention Over Configuration）”里面的最佳实践的产物，一个典型的 Spring Boot 应用本质上其实就是一个基于 Spring 框架的应用，它在 Spring 之上，构建了全新的开发模型，移除了开发 Spring 应用中很多单调乏味的内容[9]。

在 Spring 体系中，Spring Boot 是令人兴奋的新项目，它也可能改变软件开发领域的游戏规则。它提供了四个主要的特性，能够改变开发 Spring 应用程序的方式：

1）Spring Boot Starter：它将软件项目经常用到的依赖进行了统一归并，将其添加到一个依赖中，然后就可以统一添加到项目的 Maven 或 Gradle 构建中；

2）自动配置：Spring Boot 的自动配置特性继承了 Spring4 对条件化配置的支持，合理地推测软件应用所需的 bean 依赖并动态配置它们；

3）命令行接口（Command-line interface，CLI）：Spring Boot 的 CLI 发挥了 Groovy 编程语言的优势，并结合自动配置进一步简化 Spring 应用的开发[10]。

4）Actuator：它为 Spring Boot 应用添加了一定的管理特性，使监控变的简单。采用了 spring-boot-start-actuator 之后，然后直接以 rest 的方式，获取线程的运行期性能参数。

如图 3，是 Spring Boot 应用启动步骤的简要示意图，SpringApplication 作为 Spring Boot 程序启动的一站式解决方案，将一个 Spring Boot 应用启动的流程“模版化”，对开发基于 Spring 框架的微服务更加方便高效。

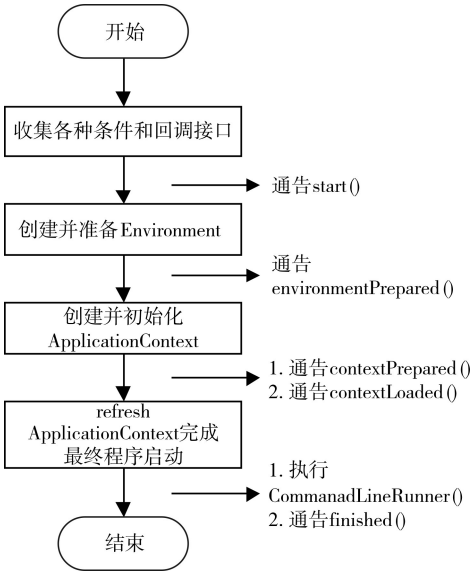

图 3　Spring Boot 应用启动步骤示意图

5　构建 Dubbo+Spring Boot 微服务的过程

新建 Maven 项目 spring-boot-starter-dubbo，将针对 Dubbo 框架的依赖以及对服务启动类的规范进行封装，此后，要开发一个基于 Spring Boot 的 Dubbo 服务，只要依赖这一自动配置模块即可。

1）项目 pom.xml 中依赖 spring-boot-starter 和 Dubbo

2）创建 ShutdownLatch 类，目的是为了阻止 Dubbo 服务的进程退出（主线程执行完毕，无其他非 daemon 线程存活）。具体做法是设置一个服务是否关闭的开关，只有当外部调用相应管理接口将服务关闭之后，再关闭当前的 Dubbo 服务。ShutdownLatch 部分代码如下：

```
if(running.compareAndSet(false,true)) {
        MBeanService  mBeanServer=ManagementFactory.getPlatformMBeanServer();
        mBeanServer.registerMBean(this, new ObjectName(domain,"name","ShutdownLatch"));
        while(running.get()) {
            TimeUnit.SECONDS.sleep(10);
        }
}
```

3）所有的 Spring Boot 应用启动都是基于标准的 SpringApplication.run 完成的，为了在启动类执行完成后可以阻止 Dubbo 服务进程的退出，需要在 Spring Boot 启动类的 main 函数最后调用 ShutdownLatch.await()，但是如果要求每个开发者在自己的

Spring Boot启动类中调用这段代码,显然这并没有给开发者的工作带来任何简化,所以本文选择使用Spring Boot中的CommandLineRunner来完成针对ShutdownLatch.await()的调用工作,任何注册到Spring Boot应用的CommandLineRunner都将在SpringApplication.run执行完成后再执行,恰好符合当前场景需要。部分代码如下:

```
public class DubboServiceLatchCommandLineRunner implements CommandLineRunner{
    @Override
    public void run(String… args) throws Exception{
        ShutdownLatch latch=new ShutdownLatch(getDomain());
        latch.await();
    }
}
```

4)定义一个JavaConfig类DubboAutoConfiguration,将DubboServiceLatchCommandLineRunner注册到Spring Boot应用的容器之中。使用@Order和@Configuration注解标注该配置类以保证DubboServiceLatchCommandLineRunner在最后执行,以避免阻塞其他逻辑的执行

5)要实现一个Spring Boot的自动配置模块,需要将上面创建的DubboAutoConfiguration配置类通过META-INF/spring.factories配置文件注册并发布。

6)以后开发Dubbo服务就只需要在服务的项目依赖中添加spring-boot-starter-dubbo依赖,然后以标准的SpringApplication加载Dubbo服务的配置并启动就可以了。

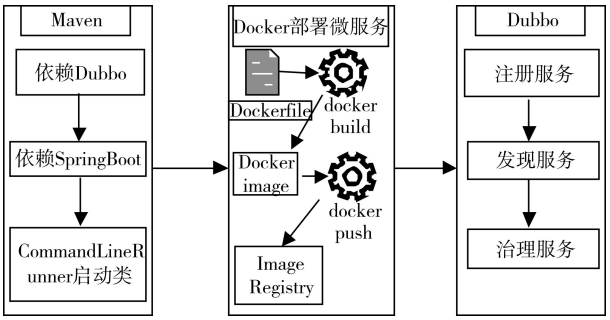

图4 Dubbo构建Spring Boot微服务流程图

如图4是基于Dubbo服务框架构建Spring Boot微服务的整体流程图。快速构建一个基于Dubbo框架的Spring Boot微服务,在Maven项目的pom.xml文件中依赖本文提出开发的spring-boot-start-dubbo,它是利用Spring Boot体系中原生的CommandLineRunner接口实现的,可以阻止Dubbo服务进程的退出。

使用Docker进行发布和部署,首先,提供一个Dockerfile用于描述Docker发布成品的构建过程,接着使用docker build命令读取Dockerfile开始构建一个Docker的image。有了docker image之后,使用docker push命令将其发布到一个Docker的image register,完成部署[15]。

6 实验测试

1)部署实验环境

采用Dockerfile创建镜像,使用容器编排工具docker-compose自动化创建基于docker容器的dubbo+zookeeper实验环境。

将dubbo-master.zip拷贝目标机解压,如图5所示。

```
# mvn clean install -Dmaven.test.skip
```

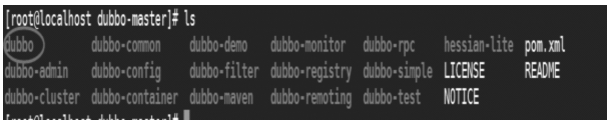

图5 dubbo在目标机解压

说明,如果整体编译失败。可以进入单个进行编译,然后在整体编译。如图6所示,Dubbo部署成功。

```
# cd dubbo
#ls
```

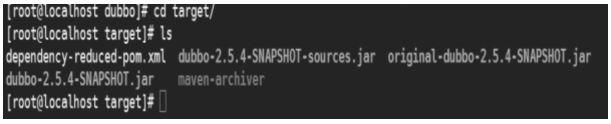

图6 Duboo编译成功

2)开发微服务

新建多个服务项目,在Maven项目pom.xml中依赖spring-boot-starter-dubbo,一键式化构建基于Dubbo框架的Spring Boot微服务。

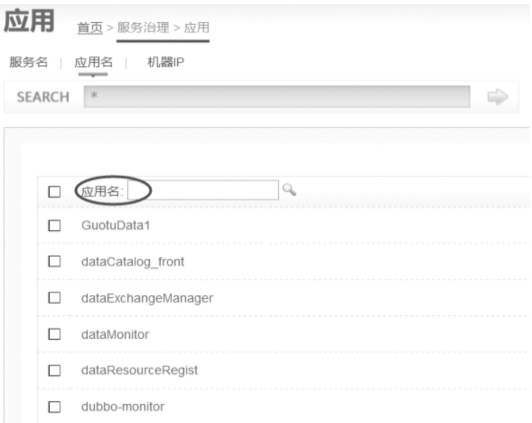

图7 微服务应用

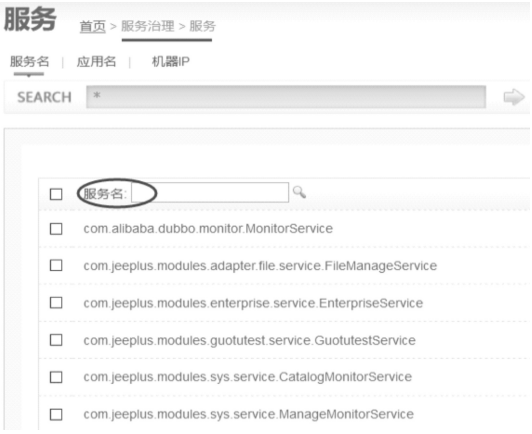

图 8　微服务 API

由图 7 和图 8 可知，服务提供者构建的多个 Dubbo+Spring Boot 微服务，可以成功运行，也可以成功被服务消费者调用。实验证明，构建多个 Dubbo+Spring Boot 微服务，具有统一接口。

7　结语

使用 Spring Boot 开发具有标准接口的微服务，Maven 在服务开发阶段进行项目管理和构建，Dubbo 提供数量众多的微服务之间的相互通信和治理。Maven+Dubbo+Spring Boot 的开发模式，使微服务的开发更加简单、逻辑清晰、标准化、规范化，让开发人员更关注业务上的功能开发，缩短软件开发生命周期。微服务在软件行业是一个新的里程碑，使服务功能模块化，将一个大的系统每一个功能单元进行拆分，本文提供了一种基于 Spring Boot 的标准化的 Dubbo 服务开发和实践，这对于海量微服务的开发和运维来说是很重要的。

参 考 文 献

[1] 邓杰文，曹彩凤. 微服务若干关键问题研究[J]. 五邑大学学报(自然科学版)，2016，30(2)：49-54.
DENG Jiewen，CAO Caifeng. Research on Some Key Problems of Microservice[J]. Journal of Wuyi University (Natural Science Edition)，2016，30(2)：49-54.

[2] 陈清金，陈存香，张岩. Docker 技术实现分析[J]. 信息通信技术，2015(2)：37-40.
CHEN QinJin，CHEN Cunxiang，ZHANG Yan. Docker technology analysis [J]. Information and Communications Technolog，2015(2)：37-40.

[3] 郭栋，王伟，曾国荪. 一种基于微服务架构的新型云件 PaaS 平台[J]. 信息网络安全，2015(11)：1671-1122.
GUO Dong，WANG Wei，ZENG Guosun. A New Cloudware Paa S Platform Based on Microservices Architecture [J]. Netinfo Security，2015(11)：1671-1122.

[4] 刘思尧，李强，李斌. 基于 Docker 技术的容器隔离性研究[J]. 软件，2015，36(4)：1003-6970.
LIU Siyao，LI Qiang，LI Bin. Research on Container Isolation Based on Docker Technology[J]. Computer engineering & Software，2015，36(4)：1003-6970.

[5] 冉勇，曹东升，杨婧孜，等. 基于 DUBBO+ZOOKEEPER 计量服务平台研究[J]. 计量与测试技术，2017(1)：1004-6941.
RAN Yong，CAO Dongsheng，YANG Jingzi，et al.DUBBO+ZOOKEEPER Measurement-Based Service Platform [J]. Metrology & Measurement Technique，2017 (1)：1004-6941.

[6] 翟金亭，吴钦卿. 基于 Dubbo 框架的视频分享系统分析[J]. 中国新通信，2016(11)：1673-4866.
ZHAI Jinting，WU Yinqing. Analysis of Video . Sharing System Based on Dubbo Framework[J]. China New Telecommunications，2016(11)：1673-4866.

[7] 陈春霞. 基于容器的微服务架构的浅析[J]. 信息系统工程，2016(3)：1001-2362.
CHEN Chunxia. Analysis of Container - based Micro Service Architecture [J]. China CIO News，2016 (3)：1001-2362.

[8] 谢璐俊，杨鹤彪. 基于 Dubbox 的分布式服务架构设计与实现[J]. 软件导刊，2016(5)：1672-7800.
XIE Lujun，YANG Hebiao. Design and Implementation of Distributed Service Architecture Based on Dubbox [J]. Software Guide，2016(5)：1672-7800.

[9] 王永和，张劲松，邓安明，等. Spring Boot 研究与应用[J]. 信息通信，2016(10)：1673-1131.
WANG Yonghe，ZHANG Jinsong，DENG Anming，et al. Research and Application of Spring Boot[J]. Information & Communications，2016(10)：1673-1131.

[10] 杨家炜. 基于 Spring Boot 的 web 设计与实现[J]. 轻工科技，2016(7)：2095-3518.
YANG Jiawei. Design and Implementation of Web Based on Spring Boot[J]. Light Industry Science and Technology，2016(7)：2095-3518.

[11] 陈涛，叶荣华. 基于 Spring Boot 和 MongoDB 的数据持久化框架研究[J]. 电脑与电信，2016(Z1)：1008-6609.
CHEN Tao，YE Ronghua. Research on Data Persistence Framework Based on Spring Boot and MongoDB [J]. Computer & Telecommunication，2016(Z1)：1008-6609.

[12] 江日念，林霞，乔德新. Maven 在 Java 项目中的引入及应用[J]. 电脑知识与技术，2013(21)：1009-3044.
JIANG Rinian，LIN Xia，QIAO Dexin. Introduction and Application of Maven [J]. Computer Knowledge and Technology，2013(21)：1009-3044.

(下转第 2551 页)

[5] 晏思宇,杨帆,黄韬. 基于OVS的SDN移动自组网络架构设计及实现[J]. 无线电通信技术,2016,42(4):69-74.
YAN Siyu, YANG Fan, HUANG Tao. Design and implementation of SDN mobile ad hoc network architecture based on OVS [J]. Radio communication technology, 2016,42(4):69-74.

[6] 杨俊东,尹强,张硕. 基于Mininet的SDN仿真与性能分析[J]. 信息通信,2017(3):189-191.
YANG Jundong, YIN Qiang, ZHANG Shuo. Simulation and performance analysis of SDN based on Mininet[J]. Information communication,2017(3):189-191.

[7] 李立耀,游莹,赵少卡,等. 基于OpenStack云平台的动态带宽分配策略[J]. 闽南师范大学学报(自然科学版),2015,28(4):47-55.
LI Liyao,YOU Ying, ZHAO Shaoka, et al. Dynamic bandwidth allocation strategy based on OpenStack cloud platform[J]. Journal of Fujian Normal University (Natural Science Edition),2015,28(4):47-55.

[8] 史律. 一种基于OpenStack的网络模型[J]. 黑龙江科技信息,2016(35):193.
SHI Lv. A network model based on OpenStack[J]. Heilongjiang Science and Technology Information,2016(35):193.

[9] 崔现东,刘江,黄韬,等. 基于节点介数和替换率的内容中心网络网内缓存策略[J]. 电子与信息学报,2014,36(1):1-7.
CUI Xiandong, LIU Jiang,HUANG Tao,et al. Cache policy in content centric network based on node betweenness and replacement rate[J]. Journal of electronics and information,2014,36(1): 1-7.

[10] 刘井莲,王大玲,赵卫绩,等. 一种面向度中心性及重叠网络社区的发现算法[J]. 计算机科学,2016,43(3):33-37,71.
LIU Jinlian, WANG Daling, ZHAO Weiji, et al. An algorithm for oriented centrality and overlay network community discovery[J]. Computer Science, 2016, 43(3): 33-37,71.

[11] 蔡宁. CDN边缘节点部署位置下沉的研究[J]. 电信工程技术与标准化,2015,28(10): 71-76.
CAI Ning. Study on placement of CDN edge nodes in deployment[J]. Telecommunication engineering technology and standardization,2015,28(10): 71-76.

[12] 张付霞,蒋朝惠. 基于DSNPP算法的社交网络隐私保护方法[J]. 计算机技术与发展,2015,25(8):152-155.
ZHANG Fuxia, JIANG Chaohui. Privacy preserving method of social network based on DSNPP algorithm[J]. Computer technology and development. 2015, 25(8): 152-155.

[13] 胡文琦,邹耀斌,雷帮军,等. 面向真皮显微镜图像的边缘空间重叠度阈值分割[J]. 微型机与应用,2016,35(15):45-47.
HU Wenqi,ZHOU Yaobing, LEI Bangjun, et al. Threshold segmentation of edge space overlap for real leather microscope images[J]. Microcomputer and Application, 2016,35(15): 45-47.

[14] 赵广松,陈鸣. 基于接收阈值的容延网络拥塞控制机制[J]. 软件学报,2013,24(1):153-163.
ZHAO Guangsong, CHEN Ming. Congestion control mechanism of delay tolerant network based on received threshold [J]. Journal of software, 2013, 24 (1): 153-163.

[15] 吴建. 融合模糊连通图和区域生长的MRI脑组织图像分割算法[J]. 科学技术与工程,2013,13(5):1135-1140.
WU Jian. MRI brain tissue image segmentation algorithm based on fuzzy connected graph and region growing[J]. Science, technology and Engineering, 2013, 13(5): 1135-1140.

[16] 王树西,李安渝. Dijkstra算法中的多邻接点与多条最短路径问题[J]. 计算机科学,2014,41(6):217-224.
WANG Shuxi, LI Anyu. Multi neighbor contact and multiple shortest path problem in Dijkstra algorithm[J]. computer science, 2014, 41(6):217-224.

(上接第2543页)

[13] 董晓光,喻涛. 使用Maven构建java项目[J]. 电子技术与软件工程,2014(10):2095-5650.
DONG Xiaoguang,YU Tao.Build a Java project using Maven[J] Electronic Technology & Software Engineering, 2014(10):2095-5650.

[14] 杨新艳,于伟涛. 基于Maven的轻量级Java软件开发研究[J]. 科技传播,2015(17):1674-6708.
YANG Xinyan,WANG Weitao. Research on the Development of Lightweight Java Software Based on Maven[J]. Public Communication of Science & Technology, 2015 (17):1674-6708.

[15] 浙江大学SEL实验室著. Docker容器与容器云[M](第2版). 北京:人民邮电出版社,2016:53-57.
Zhejiang University SEL laboratory.Docker Container & Container Cloud [M] (2nd Edition). Beijing: People's Posts and Telecommunications Press,2016:53-57.