【详解】为什么选择Spring Boot作为微服务的入门级微框架

发表于2016-05-12 16:33| 次阅读| 来源EAII企业架构创新研究院| 0 条评论| 作者许二虎

**摘要：**1. Spring Boot是什么，解决哪些问题 1) Spring Boot使编码变简单 2) Spring Boot使配置变简单 3) Spring Boot使部署变简单 4) Spring Boot使监控变简单 5) Spring Boot的不足 2. Spring Boot在平台中的定位，相关技术如何融合 1) SpringBoot与SEDA +MicroService + RESTful 2) SpringBoot与Mock 3. 采用了SpringBoot之后，技术管理应该如何进行 首先，我...

本文为普元云计算高级工程师许二虎在普元云计算架构设计群的微课堂分享。如需加入普元新一代数字化企业云平台研发设计群参与微课堂、架构设计与讨论直播，请直接回复此公众号：“加群 姓名 公司 职位 微信号”。



**1. Spring Boot是什么，解决哪些问题**

     1) Spring Boot使编码变简单

     2) Spring Boot使配置变简单

     3) Spring Boot使部署变简单

     4) Spring Boot使监控变简单

     5) Spring Boot的不足

**2. Spring Boot在平台中的定位，相关技术如何融合**

     1) SpringBoot与SEDA +MicroService + RESTful

     2) SpringBoot与Mock

**3. 采用了SpringBoot之后，技术管理应该如何进行**

首先，我们来看一下**spring boot是什么，它帮助我们解决了哪些问题**：



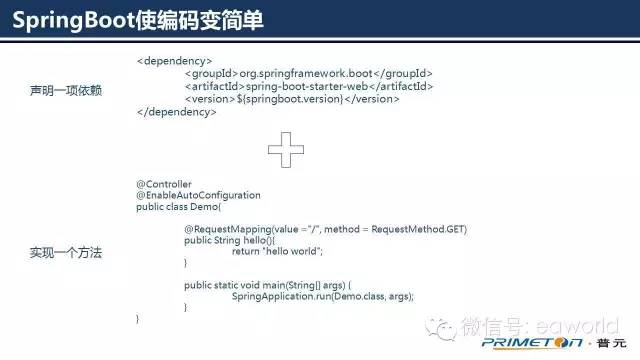
SpringBoot是伴随着Spring4.0诞生的；

从字面理解，Boot是引导的意思，因此SpringBoot帮助开发者快速搭建Spring框架；

SpringBoot帮助开发者快速启动一个Web容器；

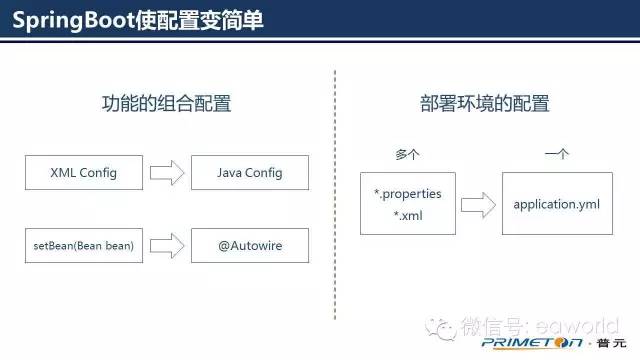
SpringBoot继承了原有Spring框架的优秀基因；

SpringBoot简化了使用Spring的过程。

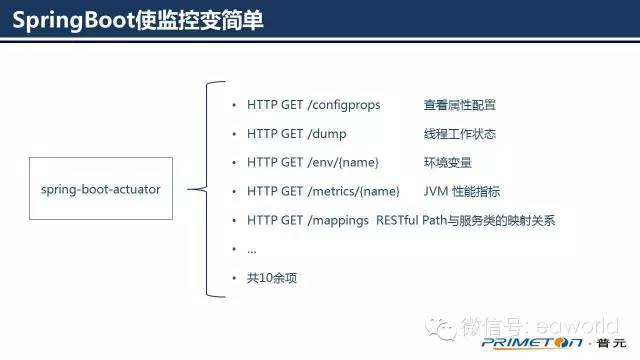


Spring由于其繁琐的配置，一度被人认为**“配置地狱”**，各种XML、Annotation配置，让人眼花缭乱，而且如果出错了也很难找出原因。

Spring Boot更多的是采用Java Config的方式，对Spring进行配置。

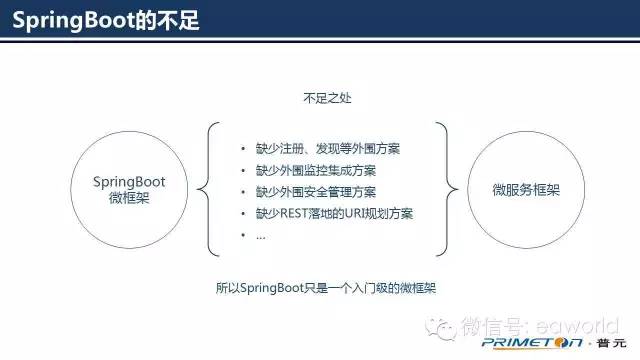






可以看到，采用了spring-boot-start-actuator之后，直接以REST的方式，获取进程的运行期性能参数。

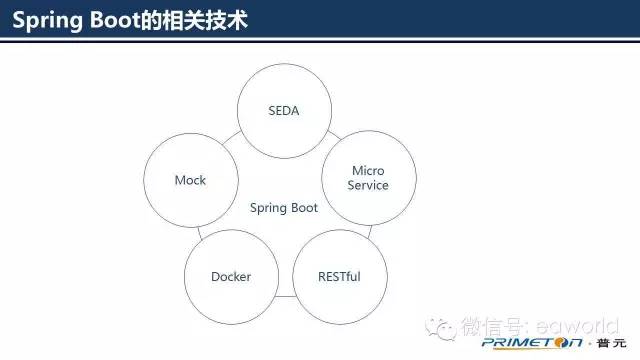
当然这些metrics有些是有敏感数据的，spring-boot-start-actuator为此提供了一些Basic Authentication认证的方案，这些方案在实际应用过程中也是不足的。

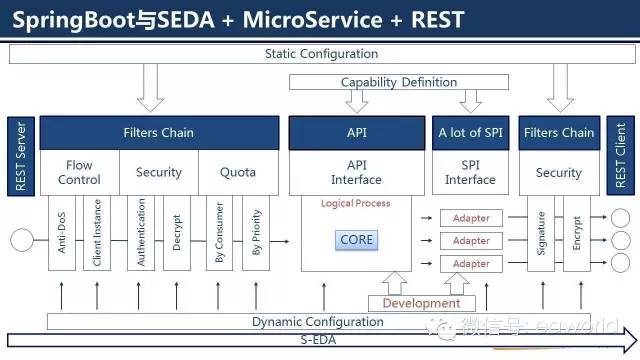


Spring Boot作为一个微框架，离微服务的实现还是有距离的。

没有提供相应的服务发现和注册的配套功能，自身的acturator所提供的监控功能，也需要与现有的监控对接。没有配套的安全管控方案，对于REST的落地，还需要自行结合实际进行URI的规范化工作。

下面，我们研究一下**Spring Boot在平台中的定位，相关技术如何融合**。



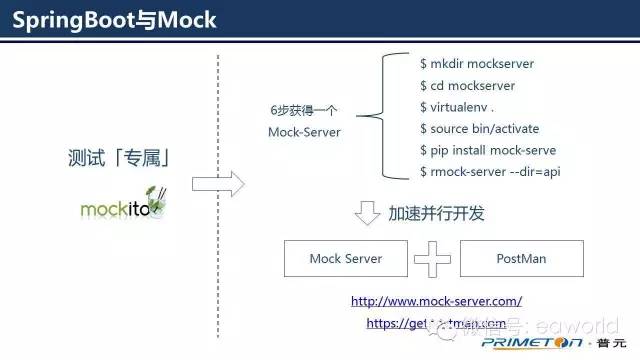


上图比较复杂，整体是采用SEDA，也就是Stage-EDA。可以看到，整体是以处理顺序进行展示的，响应过程类似。在处理过程中，主要会有前置过滤，核心功能处理，后置过滤几大部分。

图中的过滤器都是可插拔式的，并且可以根据实际场景进行扩展开发。每个过滤器都是Stage，比如ClientInstance合法性检查、调用鉴权、解密、限流等等。

一个请求Stage与Stage的转换，实现上是切换不同的线程池，并以EDA的方式驱动。

对于业务逻辑的开发者而言，只需要关心CORE部分的业务逻辑实现，其他的非功能都由框架进行统一实现。



Mock不应当再是测试的专有名词了，当然对于测试这个角色而言，mockito这样的工具，依然可以为他们提升不少效率。

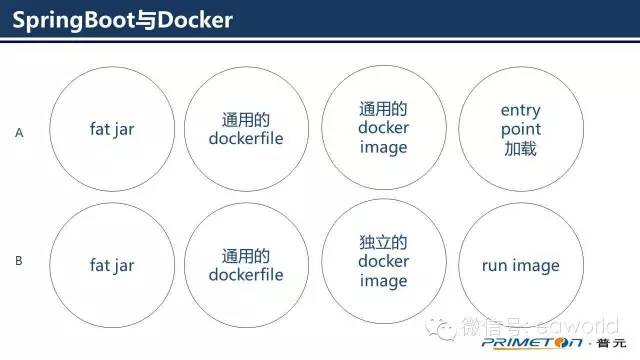
SpringBoot为创建REST服务提供了简便的途径，相比之下，采用阿里的dubbo在做多团队、多进程联调时，mock的难度就陡增。

Mock是解耦并行开发的利器，在理性的情况下，软件从开发期Mock联调，到开发与开发的真实联调，只需要切换一个依赖的域名即可，比如：

mockURI:http://mock.service.net/v1/function?param1=value1

devURI:http://dev.service.net/v1/function?param1=value1

而上述的域名切换，只需要在开发期定义好一个配置项，在做环境切换的时候自动注入即可，省时、省心、省力。



如上图和**docker的集成可以有AB两种方案**：

• A方案的核心是，把docker作为操作系统环境的交付基线，也就是不同的fat jar 使用相同的操作系统版本、相同的JVM环境。但对于docker image来说都是一样的。

• B方案的核心是，不同的fat jar，独立的编译为docker image，在启动时直接启动带有特定版本的image。

A相比与B方案的特点是对于docker registry（也就是docker的镜像仓库）的依赖性较低，对于前期编译过程的要求也较低。

采用了Spring Boot之后，**技术管理应该如何进行**？



正因为Spring Boot是与Spring一脉相承的，所以对于广大的Java开发者而言，对于Spring的学习成本几乎为零。

在实践Spring Boot时学习重点，或者说**思维方式改变的重点在于**：

1）对于REST的理解，这一点尤为重要，需要从设计、开发多个角色达成共识，很多时候都是对于HTTP 1.1协议以及REST的精髓不理解，导致REST被「盲用」而产生一些不好的效果。

2）对于YAML的理解和对于JavaConfig的理解，这两点相对较为简单，本质上是简化了xml文件，并提供等价的配置表述能力。



1. 丰富的工具链为SpringBoot的推广带来了利好。

2. SpringBoot的工具链主要来自于两个方面：

    1) 原有Spring积累的工具链；

    2) SpringMVC或者其他REST框架使用HTTP协议，使得HTTP丰富的工具成为SpringBoot天然的资源。



SpringBoot自身对于前面提到的配置文件：“application.yml”提供了多个「Profile」，可以便于开发者描述不同环境的配置，这些配置例如数据库的连接地址、用户名和密码。

但是对于企业用户而言，把不同环境的配置，写到同一个配置文件中，是极其不安全的，是一个非常危险的动作。

有一个经常被提及的例子是，随着开源的进行，很多互联网公司，都由于把相关的代码提交到github之类的开源代码社区，并且没有对代码进行严格的配置审查，导致一些”password”被公开。有些不良用心的人，就利用搜索工具，专门去挖掘这些关键字，进而导致数据库被「拖库」。

所以对于企业用户，更多的应该是采用集中式的配置管理系统，将不同环境的配置严格区分地存放。

虽然SpringBoot的actuator自身提供了基于「用户名+口令」的最简单的认证方式，但它保护的是对框架自身运行期的性能指标敏感数据的最基本的保护。这种保护在实际应用过程中，「用户名+口令」的管理是缺乏的，「用户名+口令」的安全配置过程是缺失的。

SpringBoot也不提供对于我们自己开发的功能的任何防护功能。

一般来讲，一个安全的信道（信息传输的通道），需要通信双方在进行正式的信息传输之前对对方进行身份认证，服务提供方还需要在此基础之上，对请求方的请求进行权限的校验，以确保业务安全。这些内容也需要基于SpringBoot进行外围的安全扩展，例如采用前面提到的S-EDA进行进程级别的安全管控。这些还需要配套的安全服务提供支持。

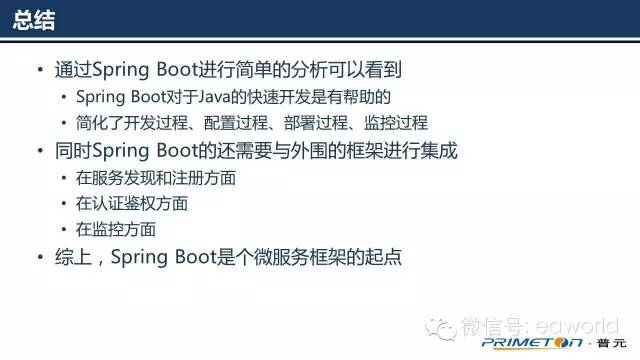
一般来说，只要企业与互联网对接，那么随便一个面向消费者的「市场活动」，就有可能为企业带来井喷的流量。

传统企业内，更多的系统是管理信息类的支撑系统，这类系统在设计时的主要用户是企业内部员工以及有限的外部供应商。这类系统存在于企业内部的时间一直很长，功能耦合也很多，在功能解耦前，是非常不适合的，或者说绝对不可以直接为互联网的用户进行服务的。

SpringBoot自身并没有提供这样的流控措施，所以需要结合前面提到的S-EDA进行流量的控制，并结合下层的水平扩展能力（例如，Kubernets）进行流量负载合理的动态扩容。

另外，在长业务流程的设计上，也尽可能地采用异步的方式，比如接口调用返回的是一个「受理号」，而不是业务的处理结果，避免井喷业务到来时，同步调用所带来的阻塞导致系统迅速崩溃，这些也都是SpringBoot自身并不解决的问题。

以上是我分享的主要内容，下面我们总结一下：



谢谢大家的聆听。



**关于作者：**

**许二虎**

EAII-企业架构创新研究院 专家委员

现任普元云计算高级工程师。毕业于中国科学技术大学，软件工程硕士。曾任职于群硕软件、科纬迅软件服务、平安健康，具备互联网领域的技术应用经验。在普元云计算平台中的角色是，以靠谱的后端的功底，支撑着整个DevOps业务平台交付。



**关于EAII**

EAII（Enterprise Architecture Innovation Institute）企业架构创新研究院，是专注于企业架构与业务创新领域的研究机构，致力于金融、电信、能源与政务等行业领域的企业软件架构优化设计与 创新研究，以及分布式计算、服务构件技术、可视化技术、业务流程管理、内存计算、企业移动计算、数据治理等领域的技术研究。