# Spring Boot

## 什么是Spring Boot

Spring Boot是一个框架，一种全新的编程规范。设计目的是用来简化新Spring应用的初始搭建以及开发过程。该框架使用了特定的方式来进行配置，从而使开发人员不再需要定义样板化的配置。

使用Spring Boot好处：

其实就是简单、快速、方便！

## Maven

Maven是一个项目构建管理工具，提供了构建、依赖、发布等。Spring Boot就是用Maven所管理。

## Maven构建项目

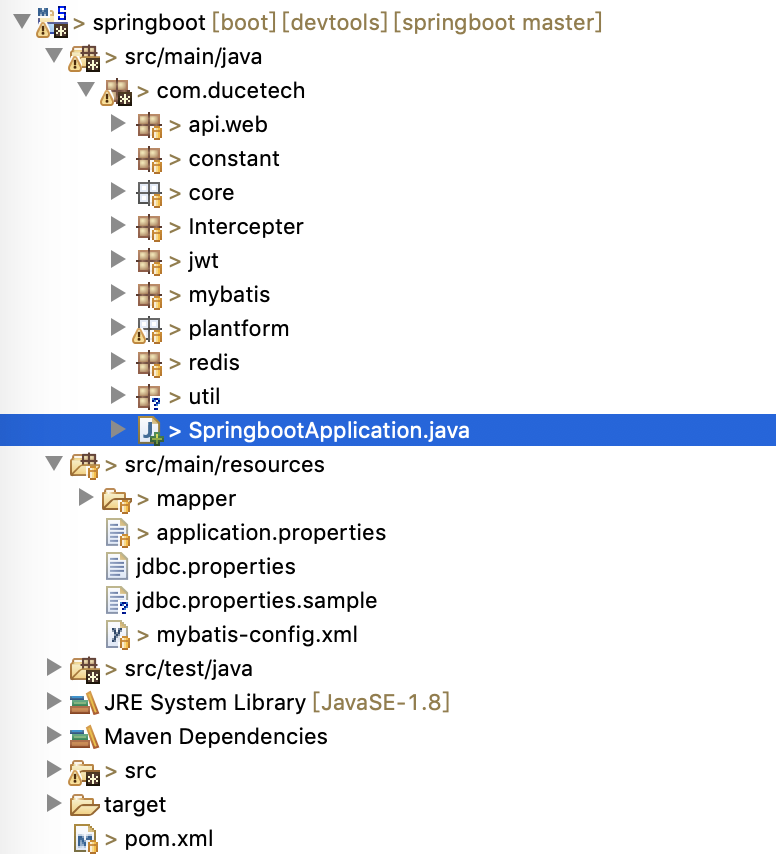
1. 访问http://start.spring.io/

2．选择构建工具Maven Project、Spring Boot版本以及一些工程基本信息，点击“Switch to the full version.”java版本建议选择1.8

3. 点击Generate Project下载项目压缩包

4. 解压后，使用eclipse，Import -> Existing Maven Projects -> Next ->选择解压后的文件夹-> Finsh，OK done!

## 项目结构介绍



1. src/main/java 程序开发以及主程序入口
2. src/main/resources 配置文件
3. src/test/java 测试程序
4. SpringbootApplication：建议放在根目录下面，主要用于一些框架配置
5. application.properties：用于配置项目运行所需的配置数据
6. pom.xml：项目maven包管理。

## 单体应用架构存在的问题

UI和若干业务模块最终都被打包在一个war包中，单体应用比较容易部署、测试，在项目的初期，单体应用可以很好地运行。然后，随着需求的不断增加，越来越多的人加入开发团队，代码库也在飞速地膨胀。慢慢地，单体应用变得越来越臃肿，可维护性、灵活性逐渐降低，维护成本越来越高。

复杂性高：整个项目包含的模块非常多、模块的边界模糊、依赖关系不清晰、代码质量参差不齐、混乱地堆砌在一起。

部署效率低：随着代码增多，构建和部署的时间也会增加。每次功能的变更或者缺陷的修复都会导致需要重新部署整个应用。

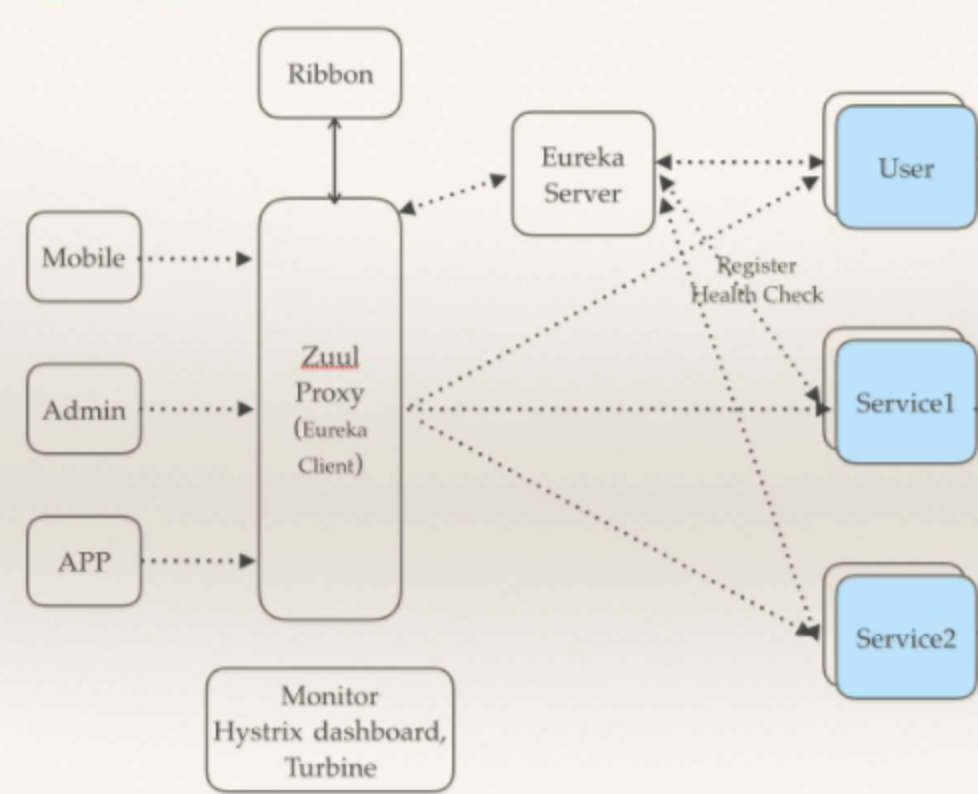
可靠性差：某个应用Bug，可能会导致整个应用的崩溃。

## 微服务

微服务风格是一种将一个单一应用程序开发为一组小型服务的方法，每个服务运行在自己的进程中，服务间通信采用轻量级通信机制（通常用HTTP资源API）。

优点：

1. 每个微服务可独立运行在自己的进程里。
2. 一系列独立运行的微服务共同构建起整个系统。
3. 每个服务为独立的业务开发，一个微服务只关注某个特定的功能，例如模型管理、物料管理。
4. 微服务之间通过一些轻量的通信机制进行通信，例如通过RESTful API进行调用。



Spring Cloud微服务

Zuul：Zuul的作用就是路由转发和过滤，即将请求转发到微服务或拦截请求，Zuul默认集成了负载均衡功能。

Ribbon：通过算法实现负载均衡功能，比如轮询等。

Eureka：注册中心。每个服务单元向注册中心登记自己提供的服务，包括服务的主机与端口号、服务版本号、通讯协议等一些附加信息。注册中心按照服务名分类组织服务清单，同时还需要以心跳检测的方式去监测清单中的服务是否可用，若不可用需要从服务清单中剔除，以达到排除故障服务的效果。

Feign：是 REST 客户端, 当你提供了 rest 服务的时候，一般返回的是 json，当使用这个api 的时候，你需要自己写请求，然后反序列化转为自己想要的 object，使用客户端的方便之处就是它帮你完成了，你使用的时候就如调用一个 java 的 interface。

Hystrix：断路器。由于网络原因或者自身的原因，服务并不能保证100%可用，如果单个服务出现问题，调用这个服务就会出现线程阻塞，此时若有大量的请求涌入，Servlet容器的线程资源会被消耗完毕，导致服务瘫痪。如果使用断路器，当断路打开后，可避免连锁故障，fallback方法可以直接返回一个固定值。

过程：

1． 多个服务与Zuul被注册到Eureka注册中心中

2． 在zuul通过加载动态过滤机制、实现请求验证、动态路由和负载分发等功能

3． 由Ribbon进行均衡负载后，分发到后端的具体服务

4． 微服务之间通过Feign进行通信处理业务

5． Hystrix负责处理服务超时熔断

## JWT（JSON Web Token）

什么是JWT：

JSON Web Tokens是一种开放的行业标准（RFC 7519），用于在双方之间安全的表述性声明规范。它将用户信息加密到token里，服务器不保存任何用户信息。服务器通过使用保存的密钥验证token的正确性，只要正确即通过验证。

优点：

1. 因为json的通用性，所以JWT是可以进行跨语言支持的，像JAVA,JavaScript,NodeJS,PHP等很多语言都可以使用。
2. 便于传输，jwt的构成非常简单，字节占用很小，所以它是非常便于传输的。
3. 因为存在客户端，减少服务器压力。
4. 通用性，JWT适应多种客户端平台，Web、APP等，尤其适应APP请求无状态的特点，具有通用性和便利性的特点。

JWT结构：



WT包含三个部分，并用点(.)连接这三部分：

1. Header
2. Payload
3. Signature

Header：

Header通常由两部分组成：类型，即JWT，以及使用的哈希算法，如HMAC SHA256或RSA，如：

{

"alg": "HS256",

"typ": "JWT"

}

然后，这个对这个json进行Base64Url编码形成JWT的第一部分。

Payoad：

第二部分是包含claim的负载，claim是关于实体（通常是用户）和附加元数据的声明，有三种类型的claim： reserved , public 和 private。

* Reserved claims：这些claim是JWT预先定义好的，并不强制使用，但建议使用，以提供一组有用的，可互操作的声明。 其中一些是：issuer（签发者），exp（到期时间），sub（面向用户），aud（接收用户）等。
* Public claims：可以根据使用需求自定义，但为避免冲突，应在IANA JSON Web令牌注册表中定义它们，或者将其定义为包含防冲突命名空间的URI。
* Private claims：这些claim是为了在使用它们的各方之间共享信息而创建的自定义claim。
* {
* "iss": "yunxiuServ",
* "aud": "www.deviceManage.com",
* "sub": "device@deviceManage.com",
* "uid": "UID\_AeId8-8UemA-kI8y7-9Loqz-jU7D4",
* "iat": 1416797419,
* "exp": 1448333419,
* }

这一部分同样经过Base64Url编码形成了JWT的第二部分。

Signature：

要创建Signature部分，您必须使用编码后的Header，编码后的Payload，密钥，Header中指定的加密算法进行签名。

例如，如果要使用HMAC SHA256算法，将以以下方式创建签名：

HMACSHA256(

base64UrlEncode(header) + "." +

base64UrlEncode(payload),

secret)

Signature用于验证JWT的发送人身份以及用于确保该消息没有经过篡改。

使用过程：

在身份验证中，服务端返回给前端和APP端的JWT必须保存在本地（Web端通常保存在LocalStorage，但也可以使用Cookie，APP通常保存在SP中），后续所有的请求过程都是无状态的，而不是传统的方法来创建Session，服务器并返回一个Cookie。

后续用户的所有请求都附带JWT，服务端用以验证请求的合法性，通常JWT信息我们放在HTTP的Request Header中。