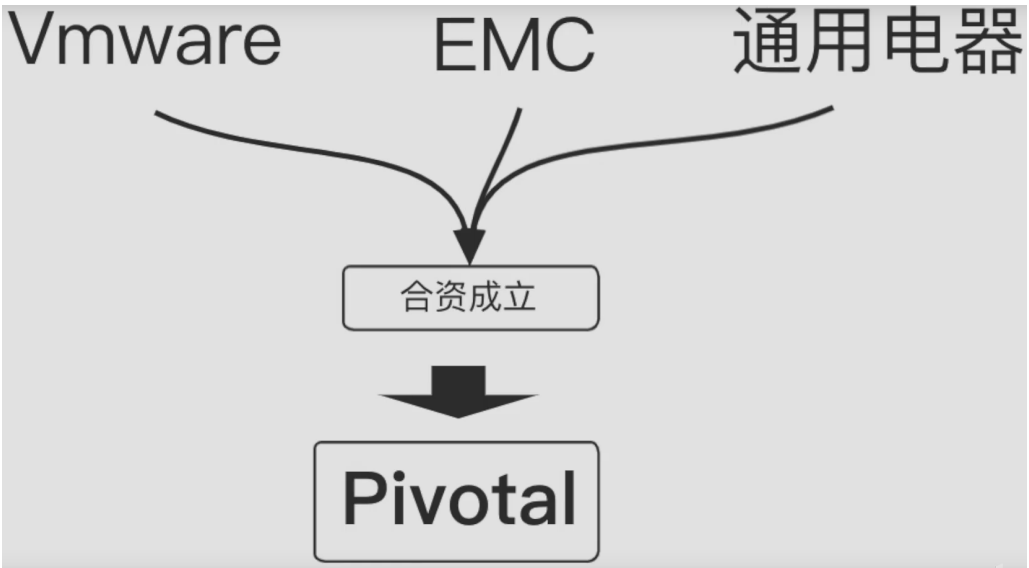
Spring Boot 简介及快速搭建

1.简介

SpringBoot它基于Spring4.0设计，是由 Pivotal 公司提供的框架。



2014 年 4 月发布 Spring Boot 1.0 基于Spring4.0

2018 年 3 月 Spring Boot 2.0发布 基于Spring 5.0。

SpringBoot 基于 Spring 开发。不仅继承了Spring框架原有的优秀特性，它并不是用来替代 Spring 的解决方案，而和 Spring 框架紧密结合进一步简化了Spring应用的整个搭建和开发过程。**其设计目的是用来简化 Spring 应用的初始搭建以及开发过程**怎么简化的呢？就是通过提供默认配置等方式让我们更容易使用。

关于 SpringBoot 有一句很出名的话就是**约定大于配置**。采用 Spring Boot 可以大大的简化开发模式，它集成了大量常用的第三方库配置，所有你想集成的常用框架，它都有对应的组件支持，例如 Redis、MongoDB、Dubbo、kafka，ES等等。SpringBoot 应用中这些第三方库几乎可以零配置地开箱即用，大部分的 SpringBoot 应用都只需要非常少量的配置代码，开发者能够更加专注于业务逻辑。另外SpringBoot通过集成大量的框架使得依赖包的版本冲突，以及引用的不稳定性等问题得到了很好的解决。

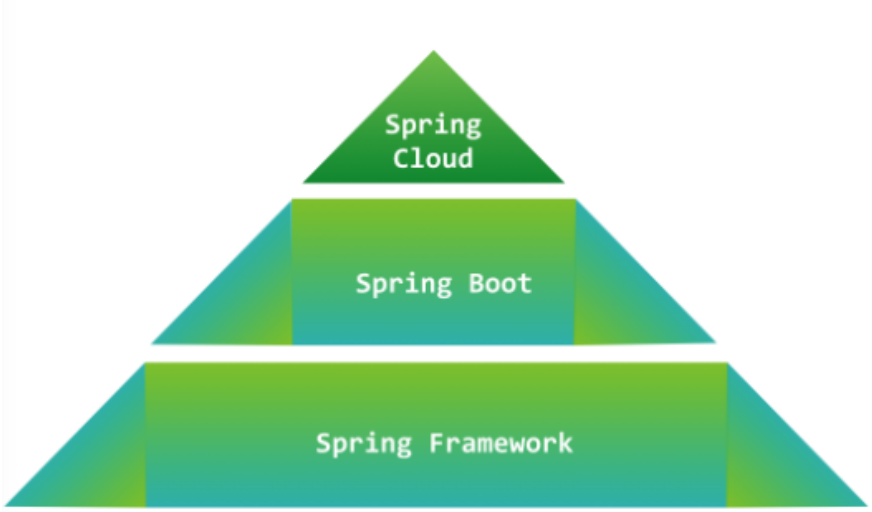
> 简化Spring应用开发的一个框架；
> 对整个企业级开发技术栈的一个大整合build anything；
> J2EE开发的一站式解决方案；

优点

--快速构建一个独立的 Spring 应用程序；
--嵌入的 Tomcat 、 Jetty 或者 Undertow，无须部署 WAR 文件；
--提供starter POMs来简化Maven配置和减少版本冲突所带来的问题；
--对Spring和第三方库提供默认配置，也可修改默认值，简化框架配置；
--提供生产就绪型功能，如指标、健康检查和外部配置；
--无需配置XML，无代码生成，开箱即用；

2.Why SpringBoot?

刚才说 SpringBoot 简化了基于 Spring 开发，这只是最直观的一方面；还有一个方面： 也更得力于各微服务组件的支持,这也是谈 SpringBoot 必谈微服务的原因。 (起初是Netflix移植到Spring),可以说是Spring Cloud带动了SpringBoot，SpringBoot成就了SpringCloud。



SpringBoot和微服务的火热程度是同步的

2.1微服务

2014年一个叫 Martin Fowler （同时也是经典著作《重构：改善既有代码的设计》一书的作者）发表了一篇关于微服务的博客，比较形象生动地介绍了什么是微服务，然后微服务才慢慢被人所熟知。

他说微服务其实是一种架构风格，我们在开发一个应用的时候这个应用应该是由一组小型服务组成，每个小型服务都运行在自己的进程内；小服务之间通过HTTP的方式进行互联互通。

和微服务相对应的就是我们之前的，单体应用，就是大名鼎鼎的 all in one 的风格。这种风格把所有的东西都写在一个应用里面，比如我们熟悉的OA，CRM，ERP系统，所有的页面，所有的代码都放在一起，打成打成一个war包，然后把war包放在Tomcat容器中运行。

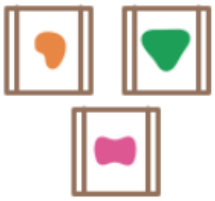
这种传统web开发的架构模式当然也有它的优势，比如它测试部署比较简单，因为不涉及到多个服务的互联互调，只需要把一个包上传到服务器就行了，可以说是一人吃饱全家不饿。同样也不会给运维带来麻烦，方便水平扩展，只需要又把相同的

应用复制多份放在不同的服务器中就达到了扩展的目的。

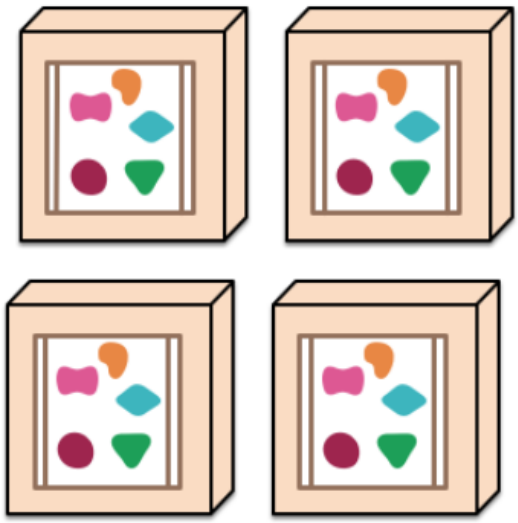
单体应用的的缺点也显而易见，容易牵一发而动全身，比如要更改一个小小的功能，就可能需要重新部署整个应用。当然，更大的挑战就是日益增长的用户需求。

一个单体应用程序把它所有的功能放在一个单一进程中...

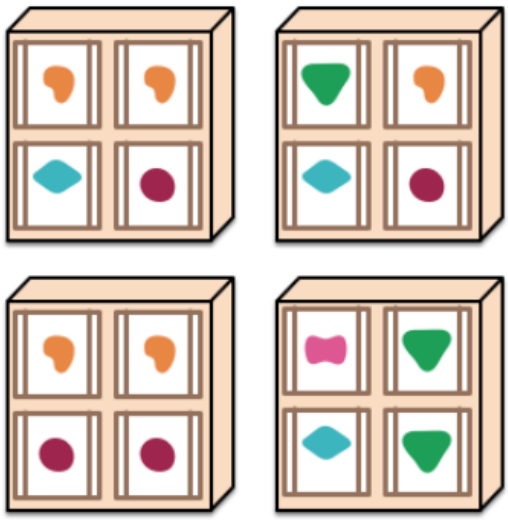
一个微服务架构把每个功能元素放进一个独立的服务中...

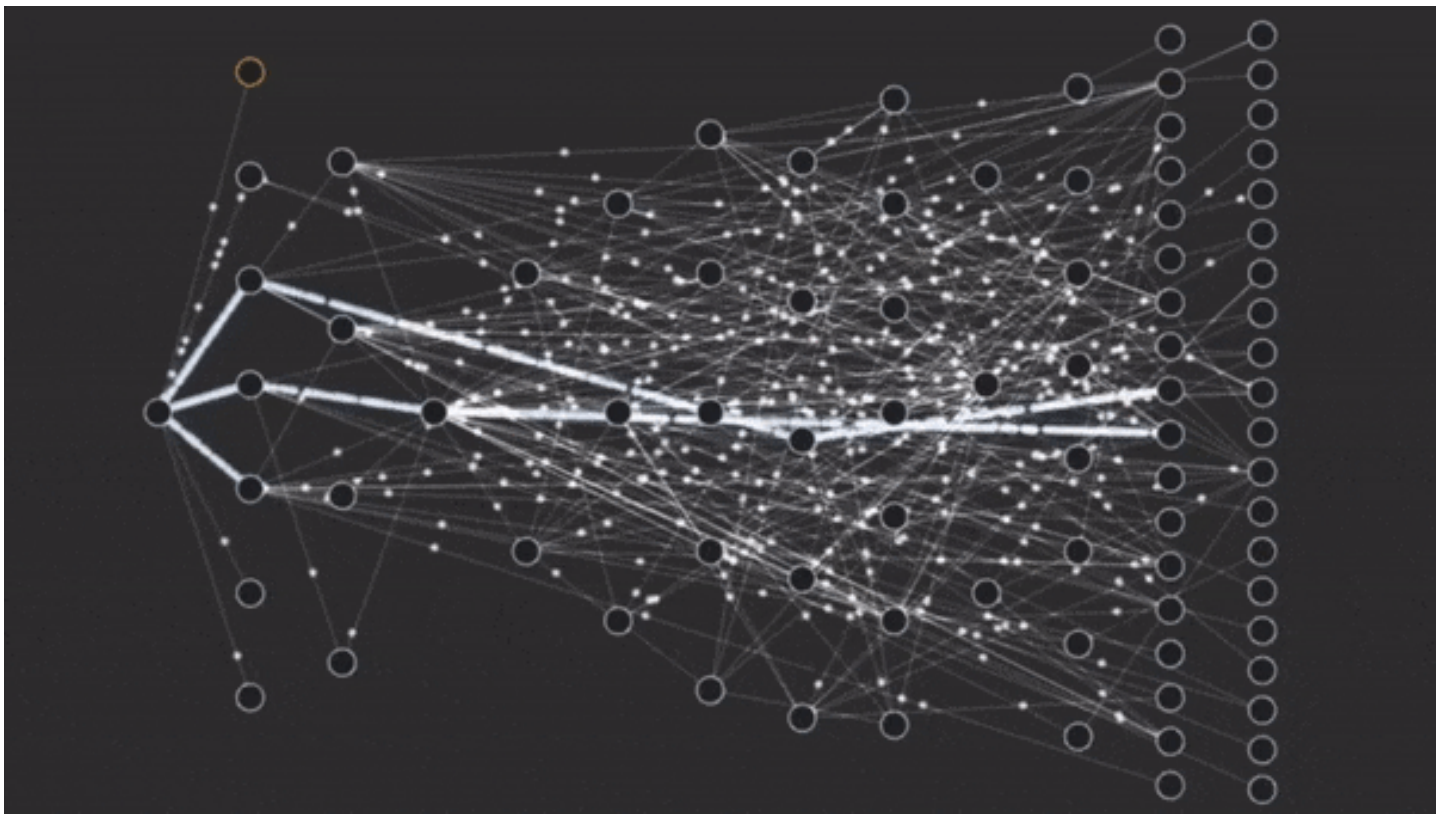


...并且通过在多个服务器上复制这个单体进行扩展



...并且通过跨服务器分发这些服务进行扩展，只在需要时才复制.





3.快速开始 SpringBoot Hello World

前置知识要求：
–SSM框架的使用经验
–熟练使用Maven进行项目构建和依赖管理
–熟练使用Eclipse或者**IDEA 2018+**

环境要求

Spring Boot 2.3.5.RELEASE需要**Java 8**，并且与Java 15（包括）兼容。 还需要Spring Framework 5.2.10.RELEASE或更高版本。

构建工具的支持

Build Tool	Version
Maven	**3.3+**
Gradle	6 (6.3 or later). 5.6.x is also supported but in a deprecated

开发工具

eclipse
Idea 2018+（别忘了配置Maven、jdk)

3.1创建一个maven工程； (jar)

3.2 项目中引入依赖

```
  <!--继承springboot的父项目-->
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.3.5.RELEASE</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>

   <dependencies>
       <!--引入springboot的web支持-->
       <dependency>
           <groupId>org.springframework.boot</groupId>
           <artifactId>spring-boot-starter-web</artifactId>
       </dependency>
   </dependencies>
```

3.3 建包并创建控制器

```
//在项目中创建指定的包结构
/*
cn
   +| tulingxueyuan
        +| controller */
@Controller
@RequestMapping("/hello")
public class HelloController {
   @RequestMapping("/world")
   @ResponseBody
   public String hello(){
        System.out.println("======hello world=======");
        return "hello";
  }
}

```

3.4 编写启动类

```
//在项目中如下的包结构中创建启动类 Application
/*
cn
   +| tulingxueyuan
         +| Application.java */
            @SpringBootApplication
            public class Application {
                public static void main(String[] args) {
                    SpringApplication.run(Application.class,args);
                }
            }
```

3.6 运行main启动项目

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.2.7.RELEASE)

2020-11-04 22:33:37.003  INFO 30180 --- [           main] com.tuling.SpringbootTestApplication     : Starting SpringbootTestApplication
2020-11-04 22:33:37.008  INFO 30180 --- [           main] com.tuling.SpringbootTestApplication     : No active profile set, falling bac
2020-11-04 22:33:38.122  INFO 30180 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8
2020-11-04 22:33:38.137  INFO 30180 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2020-11-04 22:33:38.137  INFO 30180 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache T
2020-11-04 22:33:38.139  INFO 30180 --- [           main] o.a.catalina.core.AprLifecycleListener   : An older version [1.2.16] of the A
2020-11-04 22:33:38.139  INFO 30180 --- [           main] o.a.catalina.core.AprLifecycleListener   : Loaded APR based Apache Tomcat Nat
2020-11-04 22:33:38.139  INFO 30180 --- [           main] o.a.catalina.core.AprLifecycleListener   : APR capabilities: IPv6 [true], sen
2020-11-04 22:33:38.140  INFO 30180 --- [           main] o.a.catalina.core.AprLifecycleListener   : APR/OpenSSL configuration: useAprC
2020-11-04 22:33:39.364  INFO 30180 --- [           main] o.a.catalina.core.AprLifecycleListener   : OpenSSL successfully initialized [
2020-11-04 22:33:39.573  INFO 30180 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebAp
2020-11-04 22:33:39.573  INFO 30180 --- [           main] o.s.web.context.ContextLoader            : Root WebApplicationContext: initia
2020-11-04 22:33:39.891  INFO 30180 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'appl
2020-11-04 22:33:40.079  INFO 30180 --- [           main] o.s.b.a.e.web.EndpointLinksResolver      : Exposing 2 endpoint(s) beneath bas
2020-11-04 22:33:40.140  INFO 30180 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (h
2020-11-04 22:33:40.143  INFO 30180 --- [           main] com.tuling.SpringbootTestApplication     : Started SpringbootTestApplication
2020-11-04 22:33:41.351  INFO 30180 --- [3)-192.168.56.1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServ
2020-11-04 22:33:41.351  INFO 30180 --- [3)-192.168.56.1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherSe
2020-11-04 22:33:41.356  INFO 30180 --- [3)-192.168.56.1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 5 ms
```

出现以上日志说明启动成功

3.7 访问项目

> 注意：springboot的项目默认没有项目名
>
> 访问路径：http://localhost:8080/hello/world

3.7 修改端口

如果出现Web server failed to start. Port 8080 was already in use.说明端口正在使用，我们需要修改默认端口

> 项目中src/main/resources/application.properties

```
server.port=8088
```

> 访问路径：http://localhost:8088/hello/world

3.8 部署服务器

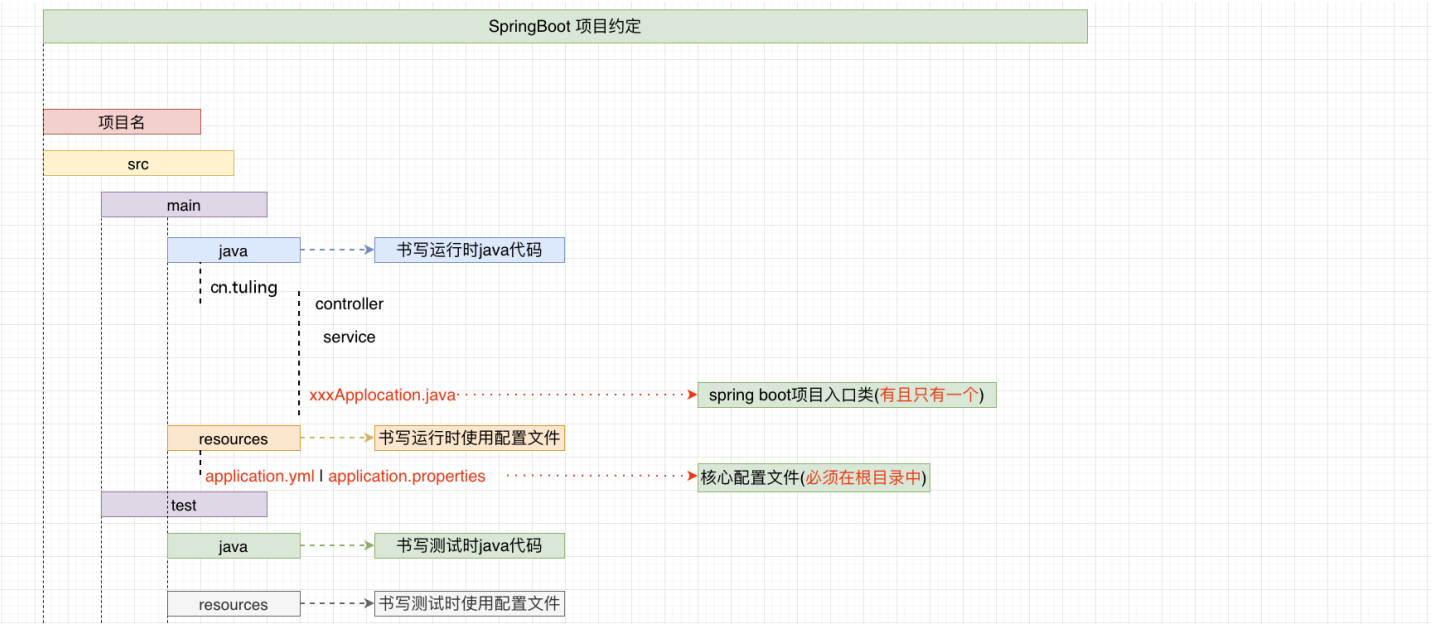
```
<!-- 这个插件，可以将应用打包成一个可执行的jar包； -->
<build>
<plugins>
    <plugin>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</artifactId>
```

```
    </plugin>
</plugins>
</build>
```

将这个应用打成jar包，直接使用java -jar的命令进行执行；

4.代码说明

文件说明



pom.xml

- **spring-boot-starter-parent**

```
<!--这是Spring Boot的父级依赖，这样当前的项目就是Spring Boot项目了。
它用来提供相关的Maven默认依赖。使用它之后，常用的包依赖可以省去version标签。-->
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.3.5.RELEASE</version>
    <relativePath/> <!-- lookup parent from repository -->
</parent>
```

```
<!--
spring-boot-starter-parent的父maven
它管理了Spring Boot应用里面的所有依赖版本;
-->
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-dependencies</artifactId>
  <version>2.3.5.RELEASE</version>
</parent>
```

- **spring-boot-starter-web**

  spring-boot-starter：spring-boot场景启动器；帮我们导入了web模块正常运行所依赖的组件；

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

Spring Boot将所有的功能场景都抽取出来，做成一个个的starters（启动器），只需要在项目里面引入这些starter相关场景的所有依赖都会导入进来。要用什么功能就导入什么场景的启动器

- **Application 启动类**

```
/**
 *  @SpringBootApplication 来标注一个主程序类，说明这是一个Spring Boot应用
    自动装配就是从这里开始的
 */
@SpringBootApplication
public class Application {

    public static void main(String[] args) {

        // Spring应用启动起来
        SpringApplication.run(Application.class,args);
    }
}
```

描述一下SpringBoot的作用

SpringBoot有哪些特性?