# Springboot学习笔记

## 微服务：

一个项目 可以由多个 小型服务构成（微服务）

## 2.spring boot可以快速开发 微服务模块

a.简化j2ee开发

b.整个spring技术栈的整合（整合springmvc spring）

c.整个j2ee技术的整合（整合mybatis redis）

## 3.目录结构resources：

static:静态资源（js css 图片 音频 视频）

templates：模板文件（模版引擎freemarker ,thymeleaf；默认不支持jsp）

application.properties： 配置文件

## spring boot内置了tomcat

并且不需要打成war再执行。

可以在appication.properties对端口号等服务端信息进行配置

spring boot将各个应用/三方框架 设置成了一个个“场景”stater，

以后要用哪个，只需要引入那个场景即可。

选完之后，spring boot就会将 该场景所需要的所有依赖 自动注入。

例如 选择 “web”,spring boot就会将web相关的依赖（tomcat json） 全部引入本项目

## 5.springboot自动装配原理

@SpringBootApplication:spring boot的主配置类

该注解包含：

@SpringBootConfiguration：

包含@Configuration，表示“配置类”:

1.该类是一个配置类

2.加了@Configuration注解的类，会自动纳入Spring 容器 （@Component）

@Configuration

public class A//表示A是一个 用于 配置的类

{

}

**@EnableAutoConfiguration**包含

**a.@AutoConfigurationPackage** ：可以找到@SpringBootApplication所在类的包 ，作用：就会将该包及所有的子包 全部纳入spring容器

**b,@Import(AutoConfigurationImportSelector.class)**:启动时，会根据META-INF/spring.factories找到相应的三方依赖，并将这些依赖引入本项目

总结：

编写项目时，一般会 对自己写的代码 以及 三方依赖 进行配置。但是spring boot可以自动进行配置：

a:自己写的代码,spring boot通过@SpringBootConfiguration自动帮我们配置；

b. 三方依赖 通过spring-boot-autoconfigure-2.0.3.RELEASE.jar中

的META-INF/spring.factories进行声明，然后通过@EnableAutoConfiguration开启使用即可

spring-boot-autoconfigure-2.0.3.RELEASE.jar包中 包含了 J2EE整合体系中 需要的依赖。

c.如何自动装配：

研究org.springframework.boot.autoconfigure.web.servlet.HttpEncodingAutoConfiguration,\

通过观察该源码 发现：

@Configuration：标识此类是一个配置类 、将此类纳入springioc容器

@EnableConfigurationProperties(HttpEncodingProperties.class)： 通过HttpEncodingProperties将编码设置为了UTF\_8 (即自动装配为UTF\_8, 如何修改改编码：通过改HttpEncodingProperties的 predfix+属性名 进行修改 [配置文件中，yml/properties])

即：该注解给了默认编码utf8，并且提供了prefix+属性名 的方式 供我们修改编码。

@ConditionalOnProperty(prefix = "spring.http.encoding", value = "enabled", matchIfMissing = true)

当属性满足要求时，此条件成立 ：要求 如果没有配置spring.http.encoding.enabled=xxx, 则成立。

\

总结：1每一个XxAutoConfiguration 都有很多条件@ConditionalOnXxx，当这些条件都满足时，

则此配置自动装配生效（utf-8）。但是我们可以手工修改改 自动装配： XxxProperties文件中的

prefix.属性名=value

2全局配置文件中的key， 来源于某个Properties文件中的 prefix+属性名

--boot通过XxAutoConfiguration实现自动装配 ，修改默认值 XxxProperties（ prefix+属性名）

如何直到 spring boot开启了哪些自动装配、禁止了哪些自动装配： application.properties中 debug=true

Positive matches列表 表示 spring boot自动开启的装配

Negative matches列表 表示spring boot在此时 并没有启用的自动装配。

## 6.配置文件

作用：spring boot 自动配置(约定，8080 ).可以使用配置文件 对默认的配置 进行修改

默认全局配置文件：

application.properties ： k=v,或行内写法(k： v，[Set/List/数组] {map,对象类型的属性}，并且 []可省，{}不能省)

application.yml : yaml ain't myarkup language ，不是一个标记文档

注意：1. k:空格v 2.通过垂直对齐 指定层次关系 3.默认可以不写引号； ""会将其中的转义符进行转义，其他不会

server:

port: 8882

path: /a/b/c

xml：是一个标记文档

<server>

<port>8882</port>

<path>/a/b/c</path>

</server>

### 6.1通过yaml给对象注入值

yaml:

student:

#name: zs

#age: 23

sex: true

birthday: 2019/02/12

绑定：

@Component //将此Javabean

@ConfigurationProperties(prefix="student")

public class Student

绑定： @ConfigurationProperties(yml/properties) @Value("xx") 二者可以互补

@ConfigurationProperties @Value

注值 批量注入 单个

松散语法 支持 不支持

SpEL 不支持 支持

JSR303数据校验 支持 不支持

注入复杂类型 支持 不支持

简单类型：(8个基本类型/String/Date)

## 7.@PropertySource：

默认会加载application.properties/application.yml文件中的数据；

例如@PropertySource(value={"classpath:conf.properties"})加载conf.properties文件中的数据；

但是，@PropertySource只能加载properties，不能加载yml

## 8.@ImportResource

spring boot自动装配/自动配置.

spring等配置文件 默认会被spring boot自动给配置好。

如果要自己编写spring等配置文件, spring boot能否识别？ 默认不识别。

如果需要识别，则需要在spring boot主配置类上 通过@ImportResource指定配置文件的路径

但是不推荐手写spring配置文件。

配置：xml配置文件，通过注解配置。

spring boot推荐时候用注解方式进行配置：写类，@Configuration @Bean ，示例：

//配置类（等价于spring.xml）

@Configuration

public class AppConfig {

@Bean

public StudentService stuService(){//<bean id="xxxxxxxxxxxxx">

StudentService stuService = new StudentService();

//

// StudentDao stuDao = new StudentDao() ;

// stuService.setStudentDao(stuDao);

return stuService;//返回值 <bean class="xxxxxxxxxxxxx">

}

}

## 9.spring boot全局配置文件中的 占位符表达式

a.随机数 ${random.uuid}等

b.引用变量值

yml中：

student:

name: ${student.user.name}

实际引用的是properties中的student.user.name=zl67

yml中：

student:

name: ${student.user.name2:无名}

## 10.多环境的切换（profile）

a. properties

默认boot会读取application.properties环境8882

多个：

application-环境名.properties

application-dev.properties8883

application-test.properties8884

如果要选择某一个具体的环境： application.properties中指定：spring.profiles.active=环境名

如果将application.properties注释掉，spring boot仍然会读取其他appilcation-环境名.properties中的配置。并且properties的优先级高于yml

b.yml

第一个环境（主环境）

server:

port: 8883

spring:

profiles:

active: dev 指定本次采用的环境

第二个环境

---

server:

port: 8884

spring:

profiles: dev 环境名

c.动态切换环境

i:通过运行参数指定环境

（1）STS(Eclipse) ：Run Configuration - Argument - program Argument

--spring.profiles.active=环境名

(2)命令行方式：

java -jar 项目名.jar --spring.profiles.active=环境名

ii:通过vm参数指定环境

STS(Eclipse) ：Run Configuration - Argument - VM

-Dspring.profiles.active=环境名

## 11.配置文件的位置

i.项目内部的配置文件：

properties和yml中的配置，相互补充；如果冲突，则properties优先级高。

spring boot默认能够读取的application.properties/application.yml，这2个文件 可以存在于以下4个地方：

file:项目根目录/config application.properties

file:项目根目录 application.properties

classpath:项目根目录/config application.properties

classpath:项目根目录 application.properties

注意:

a.如果某项配置冲突，则优先级从上往下

b.如果不冲突，则互补结合使用

配置项目名：

properties文件中

server.servlet.context-path=/boot

ii.项目外部的配置文件： (补救)

在项目Run configuration ,argumenets:

--spring.config.location=D:/application.properties

如果 同一个配置 同时存在于 内部配置文件 和外部配置文件，则外部>内部

HW.jar 运行，8881--8882

外部配置文件

通过命令行 调用外部配置文件

java -jar 项目.jar --spring.config.location=D:/application.properties

iii.项目运行参数： (补救)

在项目Run configuration ,argumenets:

--server.port=8883

通过命令行 调用外部配置文件

java -jar 项目.jar --server.port=8883

多个地方配置时，如果冲突，优先级：

命令参数（调用外部的配置文件 > 运行参数 ）>内部文件 (properties>yaml)

官网对多配置时的顺序说明：<https://docs.spring.io/spring-boot/docs/2.0.4.RELEASE/reference/htmlsingle/#boot-features-external-config>

## 12.日志

日志框架 UCL JUL jboss-logging,logback,log4j,log4j2,slf4j...

spring boot默认选用slf4j，logback

spring boot默认帮我们配置好了日志，我们直接使用即可。

日志级别：

TRACE< DEBUG< INFO<WARN< ERROR< FATAL<OFF

springboot默认的日志级别是info（即只打印 info及之后级别的信息）；也可以自定义级别：全局配置文件中logging.level.org.yq.HelloWorld=warn ,即logging.level.主配置类所在包=级别

可以通过配置 将日志信息 存储到文件中 logging.file=springboot.log 存储到了项目的根目录中的springboot.log

也可以指定 具体的日志路径：logging.file=D:/springboot.log

也可以存储到一个 文件夹中 ，logging.path=D:/log/，并且默认的文件名是spring.log

指定日志显示格式：

a.日志显示在console中

logging.pattern.console=%d{yyyy-MM-dd} [%thread] %-5level %logger{50} - %msg%n

%d:日期时间

%thread：线程名

%-5level： 显示日志级别,-5表示从左显示5个字符宽度

%logger{50} :设置日志长度 ，例如o.s.w.s.m.m.a.

%msg：日志消息

%n ：回车

b.日志显示在文件中

logging.pattern.file=%d{yyyy-MM-dd} \*\* [%thread] \*\* %-5level \*\* %logger{50}\*\* %msg%n

默认的日志格式，是在 jar包中 相应包的xml文件中进行配置。

日志的具体使用规范：官方说明https://docs.spring.io/spring-boot/docs/2.0.4.RELEASE/reference/htmlsingle/#boot-features-custom-log-configuration

## 13.springboot开发Web项目 (静态资源 html css js

new - spring starer -设置（选择 需要的场景，web）

spring boot是一个jar，因此 静态资源就不是再存放到 webapps中， 存放在哪里？

静态资源的存放路径 通过WebMvcAutoConfiguration类-addResourceHandlers()指定:/webjars/

spring boot将静态资源存入到jar包中，引入: 从Jar目录结构的webjars开始写：http://localhost:8080/webjars/jquery/3.3.1-1/jquery.js

如何自己写 静态资源，如何放到如spring boot中？ 将自己写的 静态资源->jar,同上（不推荐）；

推荐：spring boot约定： spring boot将一些目录结构 设置成静态资源存放目录， 我们的静态资源直接放入这些目录即可 。目录在哪里？ ResourceProperties类中的CLASSPATH\_RESOURCE\_LOCATIONS中设置：

{

"classpath:/META-INF/resources/", "classpath:/resources/",

"classpath:/static/", "classpath:/public/"

}

注意：在以上目录存放资源文件后，访问时 不需要加前缀，直接访问即可：http://localhost:8080/world.html

设置欢迎页：

WebMvcAutoConfiguration类中的welcomePageHandlerMapping() -->getIndexHtml() --> location + "index.html" ,即 任意一个静态资源目录中的 Index.html就是欢迎页

网站中 网页标签的Logo是固定名字 ： favicon.ico

自定义 favicon.ico ：阅读 源码得知 ：只需要将 favicon.ico文件 放入 任意静态资源目录中即可。

总结：1.通过源码发现静态资源的目录 2.用静态资源：只需要将静态资源放入 以上目录即可

3. 其他特定的文件（欢迎页、ico），只需要 根据约定（index.html favicon.ico） 放入该目录即可

如何自定义静态资源目录（Properties文件中的 prefix+属性） ：

spring.resources.static-locations=classpath:/res/, classpath:/img/

以上就将 静态资源目录设置为了classpath:/res/, classpath:/img/ ，注意 自定义静态资源目录后 以前默认的目录会失效

动态资源： JSP(spring boot默认不支持)

推荐：模板引擎 thymeleaf

网页= 模板+数据

引入thymeleaf：到官网查询 thymeleaf的依赖（Maven）

使用thymeleaf:代码在哪里写？

ThymeleafAutoCongifutation 、

XxProperties

通过ThymeleafProperties源码得知：

使用thymeleaf只需要将 文件放入目录："classpath:/templates/"; 文件的后缀： ".html";

注意：在以前传统的web项目中：静态资源修改后 是不需要重启的；但是在spring boot项目中，修改后 需要重启。

<p th:text="${welcome}">welcome to thymeleaf....</p>以上，先从${welcome}中取值，如果有 则直接显示；如果没有，则在显示welcome to thymeleaf....

th就是替换原有html的值：th:html属性名=值 ;

<p id="pid" class="pclass" th:id="${welcome}" th:class="${welcome}" th:text="${welcome}">welcome to thymeleaf....</p>

th:xx (参见第10章 Attrubite Pre....)

th:utext 获取文本值 显示 将hello 渲染为h1后的效果

th:text 获取文本值(不转义) 显示<h1>hello</h1>

符号

th:text="${welcome}" ，除了$以外 其他符号？ 查看第四章 Standard Express....