1. 环境配置
2. 目录结构resources

Static：静态资源（js css 图片 音频 视频）

Templates：模板文件（模板引擎freemarker、thymeleaf;默认不支持jsp）

application.properties:配置文件

1. 内置了tomcat，并且不需要打成war在执行

可以在application.properties对端口号等服务信息端进行配置

1. Springboot将各个应用/三方框架 设置成一个个场景stater

以后要用哪个。只需要引入那个场景即可

选完之后，就会将该场景所需要的所有依赖自动注入

如：“web” springboot就会将web相关的依赖（tomcat json）全部引入

1. @SpringBootApplication ：spring boot的主配置类
   1. @SpringBootConfiguration

包含@Configuration 表示“配置类”

1. 表示该类是一个配置类

@Configuration

Public class A{} //表示A是一个用于 配置的类

2.加了@Configuration注解的类，会自动纳入Spring容器

* 1. @EnableAutoConfiguration： 自动配置

@AutoConfigurationPackage

约定由于配置

可以找到SpringBootApplication所在类的包，作用：将改包及所有子包全部纳入spring容器

例如com.example.demo01.controller及com.example.demo01.controller

。xxx

* 1. @Import(AutoConfigurationImportSelector.class)
* 自己写的
* 第三方依赖（jar、配置）

Springboot 在启动时，会根据META-INF/spring.factories找到相应的第三方依赖，并将这些依赖引入项目

编写项目，会将自己写的代码以及三方依赖进行配置但是springboot自动配置

* 自己写的代码@SpringBootConfiguration自动帮我配置
* 三方依赖spring-boot-autoconfigure-2.1.5.RELEASE.jar/METAINF/spring.factories进行申明，然后@EnableAutoConfiguration开启使用

HttpEncodingAutoConfiguration

@Configuration 表明是一个配置类、将此类纳入springioc容器

@EnableConfigurationProperties(HttpProperties.class)通HttpProperties

private final Encoding encoding = new Encoding()设置utf-8

修改 通过HttpProperties prefix

spring.http charset

server.port=8888  
server.servlet.context-path=/boot  
spring.http.charset=iso-8859-1

@ConditionalOnWebApplication(type = ConditionalOnWebApplication.Type.*SERVLET*)  
@ConditionalOnClass(CharacterEncodingFilter.class)  
@ConditionalOnProperty(prefix = "spring.http.encoding", value = "enabled",  
 matchIfMissing = true)

第三个 当属性满足要求时候，条件成立 prefix value matchIfMissing

满足三点，配置自动生效

即每一个xxxxAutoConfiguration都有许多条件@ConditionalOnXXX条件满足是自动装配生效（utf-8）但是我们可以手工修改，prefix.属性名=value

全局配置文件中的key,来源与么某个Properties中的prefix

如何知道springboot开启了那些自动装配，禁止那些自动装配

debug=**true**

控制台显示开启Positive matches:

未开启Negative matches:

1. 配置文件

application.properties/ application.yml

作用：spring boot 自动配置（约定8080）可以使用配置文件，对默认修改

默认全局配置文件

application.properties

student.name = ls

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

<dependency>  
 <groupId> org.springframework.boot </ groupId>  
 <artifactId> spring-boot-configuration-processor </ artifactId>  
 <optional> true </ optional>  
</ dependency>

可以默认不写引号：“”会转义，其他不会

server:

port: 8888

path: a/b/v

* yml格式 key:空格value

需要缩进

xml:是一个标记文档

<server>

<port>8888</port>

</server>

多种

<server>

<port>8888</port>

<path>a/b/v</path>

</server>

**student**:  
 **name**: wcy  
 **age**: 23  
 **sex**: true  
 **birthday**: 2019/02/12  
 **location**: {**province**: 江苏, **city**: 连云港, **zone**: 赣榆} //行内写法

**location**:

**province**: 江苏

**city**: 连云港,

**zone**: 赣榆  
 **hobbies**:  
 - 足球  
 - 篮球

[足球，篮球]  
 **skills**:

[编程，金融]  
 - 编程  
 - 金融  
 **pet**:  
 **name**: cc  
 **strain**: 二哈

{ **name**: cc，**strain**: 二哈}

中括号可以省，大括号不能省

1. 绑定

方法一：

@Component //将JavaBean放入spring容器

@ConfigurationProperties(prefix = "student")引入以上两种的配置寻找

方法二：

@Value("111")  
private String name;  
@Value("333")  
private int age;

全部放入

以上两种方法可以同时使用，方法一的优先级高于方法二

|  |  |  |
| --- | --- | --- |
|  | @ConfigurationProperties | @value |
| 注值 | 批量注解 | 单独 |
| 松散语法nickName<==>nick-name | 支持 | 不支持 |
| SpringEL | 不支持 | 支持  student:  # name: wcy  uname: wwwww  @Value("${student.uname}") |
| JSR303数据校验 | 支持  @Validated//jsr303数据校验开启注解  public class Student  @Email private String email; | 不支持 |
| 注入复杂类型 | 支持  简单类型：8个基本类型/String/Date | 不支持 |

1. @PropertySource:默认加载application.properties/ application.yml文件中的数据

例如将application.properties修改conf.properties,则相关值无效，添加一些代码

@PropertySource({"classpath:conf.properties"})

手动添加，只能加载properties，无法使用yml

1. @ImportResource

Springboot 自动装配/自动配置

Spring等配置文件，默认会被springboot自动配置好

如果要自己编写spring等配置文件，springboot能够识别？

默认不识别，如果需要，通过@ImportResource

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（推荐）

1. 通过注解配置（推荐）

写类，@Configuration @Bean

@Configuration  
public class AppConfig  
{  
 @Bean  
 public StudentService stuService()  
 {  
 StudentService stuService = new StudentService();  
 // StudentDao studentDao = new StudentDao();  
// stuService.setStudentDao(studentDao);  
 return stuService;  
 }  
}

1. Spring boot 全局配置文件中占位符表达式
2. 随机数${random.int}之类的
3. 应用变量值

在application.peoperties 中 student.user.name=zl

Yml中

**student**:  
 **name**: ${**student.user.name**}

**name**: ${**student.user.name2：无名**}如果不存在2，则用默认值

1. 多环境的切换profile

开发、测试、实施.

1. properties

默认boot 会读取application.properties

多个：application-环境名.properties

Application-dev.peoperties

Application-test.peoperties

切换spring.profiles.active=dev

如果将application.properties注释掉，会寻找其他的properties，且优先级高于yml

1. yml(第一个就是主环境)

server:

port: 8888

spring:

profiles:

active: dev

---

server:

port: 8884

spring:

profiles: dev

---

server:

port: 8885

spring:

profiles: test

1. 动态切换
   * Idea Program arguments命令行方式 --spring.profiles.active=dev
   * 命令行方式：

Java -jar 项目名.jar --spring.profiles.active=dev

* + 通过jvm参数指定 VM option

-Dspring.profiles.active=dev

1. 配置文件的位置

项目内部的配置文件

Properties默认加载application.properties/ application.yml

相互补充，冲突则properties优先级高

这2个文件存在4个地方

File:项目根目录/config

File:项目根目录

classpath:项目根目录/config

classpath:项目根目录

项目目录优先级依次往下，同覆盖异互补

项目外部的配置文件

配置项目名server.servlet.context-path=/boot

<http://localhost:8881/boot/hello>

* Idea Program arguments命令行方式

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

外部>内部

* 打成jar包，个别修改

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

* idea命令行参数--server.port=8888
* Cmd java -jar 项目.jar -- server.port=888

优先级

命令参数(运行参数>外部配置文件)>运行参数>内部文件（properties>yml）

1. 日志

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

Springboot logback slf4j

Springboot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
* 默认级别info级别及之后的信息
* 自定义级别，在全局配置文件修改logging.level.com.example.demo01=warn

logging.level.com.主配置类所在包=级别

* 生成文件logging.file=springboot.log 相对于项目路径
* 可以配置绝对路径
* 文件logging.file=D:/WorkSpace/SpringBoot/LogginTest/springloggin.log
* 文件夹logging.path=D:/WorkSpace/SpringBoot/LogginTest/ 系统取名 spring.log
* 指定日志显示格式
  + 控制台

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

* + 文件

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

1. Springboot开发web项目

Springboot是一个jar 因此静态资源不是在存放到webapps中，存放在？

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

Webjars <https://www.webjars.org/>

* <dependency>

<groupId>org.webjars</groupId>

<artifactId>jquery</artifactId>

<version>3.4.1</version>

</dependency>

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？

private static final String[] CLASSPATH\_RESOURCE\_LOCATIONS = {

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

"classpath:/static/", "classpath:/public/" };

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

prefix.属性名=value

全局配置文件中的key,来源与么某个Properties中的prefix

@ConfigurationProperties(prefix = "spring.resources", ignoreUnknownFields = false)

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:text 获取文本值 显示 将hello 渲染为h1后的效果

th:utext 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....

16.Spring boot整合JSP开发

之前spring boot默认 自带一个内置的tomcat，不需要打war包，直接通过Jar即可运行。

但是，如果要整合jsp开发，就需要 单独配置一个 外置的tomcat ，需要打war包。

Spring boot整合JSP开发步骤：

1.新建boot项目， war

注意：

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-tomcat</artifactId>

<scope>provided</scope>

</dependency>

provided：意思是 将项目打包时，不需要将内置的tomcat一起打包。

2.建立基本的web项目所需要的目录结构

webapps/WEB-INF(需要)

webapps/WEB-INF/web.xml (不需要)

webapps/index.jsp

3.创建tomcat实例、部署项目

访问：

域名：端口/项目名/文件名

http://localhost:8080/SbJSP/index.jsp

分析：

如果是一个war包的spring boot项目，在启动服务器tomcat时， 会自动调用ServletInitializer类中 的configure方法，configure方法会调用spring boot的主配置类 从而启动spring boot;

即在启动tomcat服务器时 会1启动tomcat 2启动spring boot