# 原理初探

自动配置

Pom.xml

Spring-boot-dependicies:核心依赖在父工程中

我们在写或者引入一些springboot依赖的时候，不需要指定版

本因为有这些版本仓库

启动器

主程序

springboot所有自动配置类在启动的时候扫描并加载：spring.factories所有的自动配置类都在这里，但比一定生效。要判断条件是否成立，要有对应启动器

1. springboot在启动的时候，从类路径下/META-INF/spring.factories获取指定的值
2. 将这些自动配置导入容器，自动配置就会生效，帮我们进行自动配置
3. 以前我们需要自动配置的东西，现在springboot帮我们做了
4. 整合JavaEE，解决方案和自动配置都在spring-boot-autoconfigure-2-2-0这个包下
5. 它会把所有需要导入的组件，已类名的方式返回，这些组件就会被添加到容器
6. 容器中会存在非常多的xxxAutoConfiguration的文件(@Bean)，就是这些给容器中导入了这个场景需要的所有组件,并自动配置,@Configuration,javaConfig
7. 有了自动配置类就免去了手动配置

SpringApplication类

这个类做了4件事

1. 推断应用的类型是普通项目还是web项目
2. 查找并加载所有可用初始化器，设置到initializers属性中
3. 查找出所有应用程序监听器，设置到listeners属性中
4. 推断并设置main方法的定义类，找到运行的主类

SpringBoot使用一个全局的配置文件，配置文件名称是固定的

application.properties

语法结构:key=value

application.yaml

语法结构：空格value

配置文件的作用：修改SpringBoot自动配置默认值，因为SpringBoot在底层都给我们配置好了

yaml语法：

server：

Port: 8080

xml语法：

<server>

<port>8080<port>

</server>

properties语法：

Port=8080

yaml可以存对象，数组，但是properties只能存键值对，

Yaml对空格的要求非常高

Yaml可以给实体类赋值

结论：

配置yaml和propertie都可以获取到值，推荐yaml

如果我们在某个业务中，只需要获取配置文件的某个值，可以@value

如果我们专门编写一个JavaBean来和配置文件进行映射，就可以直接使用@configuProperties，不要犹豫

@configuProperties @value

功能 批量注入配置文件中的属性 一个个指定

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

复杂类型封装 支持 不支持

多套环境

1，.properties

application.properties

application-dev.properties

application-test.properties

springboot的多环境配置：可以选择激活那一个配置文件

spring.profiles.active=dev或者spring.profiles.active=test

2,.yaml

server:

port:8081

--

server:

port:8082

spring:

profiles:dev

---

server:

port:8083

spring:

profiles:test

配置文件能写的都是spring.factories里面的

xxxAutoConfiguration:默认值 xxxProperties和配置文件绑定，我们就可以使用自定义的配置

自动装配的原理：

1. SpringBoot启动会加载大量的自动配置类
2. 我们看我们需要的功能有没有在SpringBoot默认写好的自动配置类当中
3. 我们再来看这个自动配置类中到底配置了那些组件（只要我们要用的组件存在其中，我们就不需要手动配置了）
4. 给容器中自动配置类添加组件的时候，会从properties类中获取某些属性，我们值需要在配置文件中指定之恶写属性即可

xxxAutoConfiguration：自动配置类；给容器中添加组件

xxxProperties：封装配置文件中相关属性

SpringBoot Web 开发

SpringBoot到底帮我们配置了什么？

-xxxAutoConfiguration..向容器中自动配置组件

-xxxProperties:自动配置类装配文件中自定义的一些内容

要解决的问题：

1. 导入静态资源
2. 首页
3. Jsp,模板引擎
4. 装配扩展SpringMVC
5. 增删改查
6. 拦截器
7. 国际化

静态资源

总结：

在springboot中我们可以使用以下方式处理静态资源

1. Webjars localhost:8080/webjars/
2. Public,static,/\*\*,resource localhost:8080/

Resources文件夹下面的resource,public,static文件夹里的文件都可以访问，优先级resource>static>public，默认使用static

在template目录下的所有页面，只能通过controller来跳转，这个需要模板引擎的支持，thymeleaf

首页如何定制

在static中建一个index.html即可

1,所有页面的静态资源都需要使用thymeleaf接管

<html lang="en" xmlns:th="http://www.thymeleaf.org">

2，url:@{}

页面国际化

i18n:internationalization ; i(18个字母)n

注意点：

1. 我们需要配置i8n文件
2. 我们如果需要在项目中进行按钮自动切换，我们需要定义一个组件LocaleResolver
3. 记得将自己写的组件配置到spring容器中 @Bean
4. #{}

登录拦截器

增删改查

如何搭建一个网站

1, 前端：页面长什么样

模板：

框架：组件，自己手动拼接组合，Bootstrap,Layui,semantic-ui

栅格系统，导航栏，侧边栏，表单

2，设计数据库(难点！)

1. 前端让他能够独立运行，独立化工程
2. 数据接口如何对接：json，对象 all in one
3. 前后端联调测试
4. 有一套自己熟悉的后台模板：工作必要
5. 前端界面：至少自己能够通过前端框架，组合出来一个网站页面

-index -about -blog -post -user

Data

对于数据访问层，无论是SQL(关系型数据)还是NOSQL(非关系型数据)，SpringBoot底层都是采用SPringl Data的方式进行统一处理。

Mybatis

Mybatis-spring-boot-start

1. 导入包
2. 配置文件
3. mybatis配置
4. 编写sql
5. 业务层调用dao层
6. controller层调用service层

SpringSecurity