springboot小总结

https://dwz.cn/P1N121RT----狂神笔记地址

https://home.cnblogs.com/u/hellokuangshen---狂神博客笔记

1.springboot和spring的关系，有那些优缺点

- 概念：springboot 是简化spring应用开发的一个框架，整合多方技术栈的框架
- 快捷键：ctrl+alt+u 查看类图，----选中类:f4跳到源码
- 关系：

springboot和spring的关系，就像spring和jdk的关系，boot的底层就是大量使用了spring，当然我们是基于 spring4.0x以上的版本,所以说对spring了解多精通，决定着boot可以走多远。

- springboot的优点：

springboot简化了spring的配置，严格遵守约定大于配置大于硬编码，也就是说，boot底层做了大量的自动配置，只要我们满足它的配置条件，我们就不许需要手动配置，快速整合；依赖管理,springboot很好的做好了依赖管理，减少了我们在spring阶段各种依赖冲突问题，需要排查各种依赖之间的关系；内嵌tomcat，底层为我们配置好了，web自动配置类，我们需要满足它的条件，即可使用；支持各种微服务组件；部署容易。

- 缺点：

springboot这个产品技术更新迭代很快，需要我们程序员不停的去学习，出错之后排错比较困难，因为它是约定俗称好的，你违反了约定，就容易报错，不像我们自己手写的错误可以一步一步调试，不

过还好，报错信息有很好的提示。

2.springboot底层的注解

- @configuration　5.0x之后:proxyBeanMethods=true代表从ioc中找，反之自己创建
- @Import　三种注入bean的方式,直接引入class;实现importselect接口，返回数组；实现ImportBeanDefindregister;直接注册bean
- @conditional　条件装配：@conditionalOnMissingBean　没有bean；@conditionalBeanClass有bean的时候
- @ImportResource类注解：相当于可以使用xml的配置　一般不用
- @configurationProperties(prefix="mycar")　元数据　@componet　配合在一块　+@value
- @EnableConfigurationProperties(car.class)　+ @configurationProperties(prefix="mycar")+@value

3.springboot的自动装配原理

略：自己脑子中，步骤太多，不做解释

4.元数据

YAML | 复制代码

```yaml
# 应用名称
spring:
  application:
    name: maven-child-boot
mycar:
  name: guihaole
  price: 100.56
  strList: ["java","boot","spring"]
  user:
    username: guigege
    password: 1234564
  users: [{username: gui,password: 123},{username: gui1,password: 12222}]
  userMap: {success: 123,error: 123}
  userMap2: {success: {username: gui,password: 123},error: {username: gui,password: 123}}


  // 也可以指定其他的yml文件初始化----其实是spring里面的注解
@PropertySource("classpath:user.yml")
public class User {
    @Value("${username}")
    private String username;
    @Value("${password}")
    private String password;
}

```

Java | 复制代码

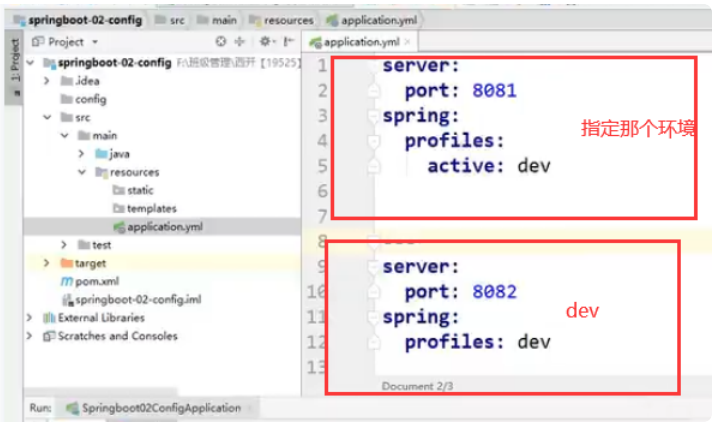
```java
@ConfigurationProperties(prefix = "mycar")
@Component
public class MyCar {
    private String name;
    private Double price;
    private List<String> strList;
    private User user;
    private List<User> users;
    private Map<String, String> userMap;
    private Map<String, User> userMap2;
}
//主启动类
//@EnableConfigurationProperties(MyCar.class)
@SpringBootApplication
public class ApplicationSpringBoot {
    public static void main(String[] args) {
        SpringApplication.run(ApplicationSpringBoot.class,args);
    }
}
```

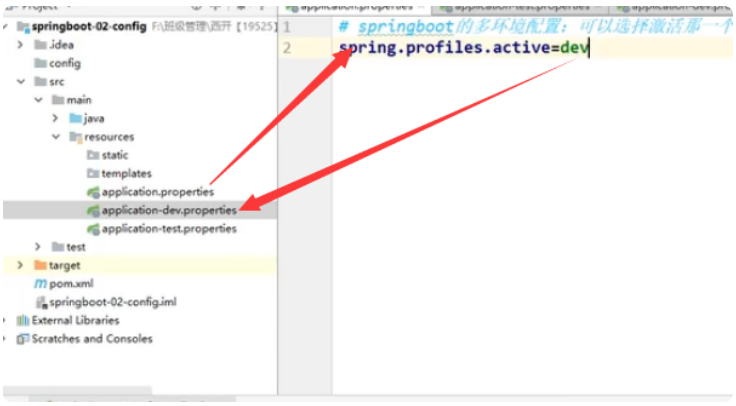
5.自定义banner.txt

略 ---- easy

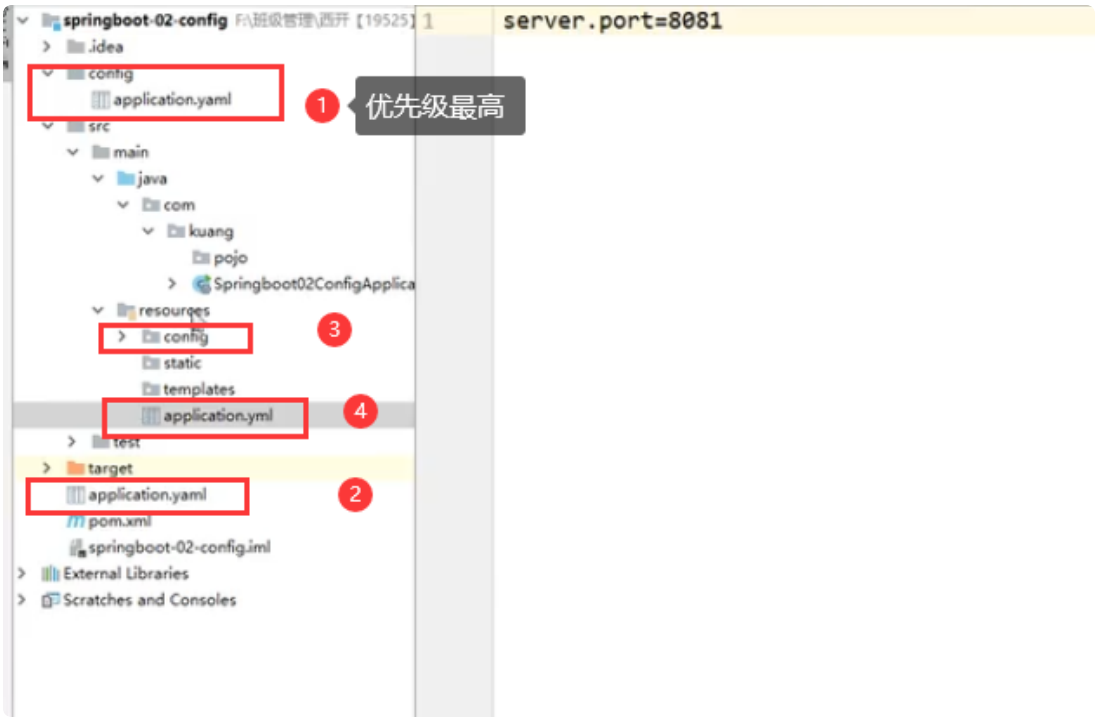
6.yml相关配置

- 多环境配置生效





- yml不同地方优先级顺序



7.JSR303数据校验

Springboot中可以用@validated来校验数据，如果数据异常则会统一抛出异常，方便异常中心统一处理。我们这里来写个注解让我们的name只能支持Email格式；

Java 复制代码

```java

@Component //注册bean
@ConfigurationProperties(prefix = "person")
@Validated  //数据校验
public class Person {

    @Email(message="邮箱格式错误") //name必须是邮箱格式
    private String name;
}
```

运行结果： default message [不是一个合法的电子邮件地址];

使用数据校验，可以保证数据的正确性；

Java | 复制代码

```java
@NotNull(message="名字不能为空")
private String userName;
@Max(value=120,message="年龄最大不能查过120")
private int age;
@Email(message="邮箱格式错误")
private String email;

空检查
@Null       验证对象是否为null
@NotNull    验证对象是否不为null, 无法查检长度为0的字符串
@NotBlank   检查约束字符串是不是Null还有被Trim的长度是否大于0,只对字符串,且会去掉前后空格.
@NotEmpty   检查约束元素是否为NULL或者是EMPTY.

Booelan检查
@AssertTrue     验证 Boolean 对象是否为 true
@AssertFalse    验证 Boolean 对象是否为 false

长度检查
@Size(min=, max=) 验证对象（Array,Collection,Map,String）长度是否在给定的范围之内
@Length(min=, max=) string is between min and max included.

日期检查
@Past       验证 Date 和 Calendar 对象是否在当前时间之前
@Future     验证 Date 和 Calendar 对象是否在当前时间之后
@Pattern    验证 String 对象是否符合正则表达式的规则

.......等等
除此以外，我们还可以自定义一些数据校验规则
```

8.springboot如何绑定application.yml

springboot通过启动类，根据自动配置会为我们加载很多符合条件的autoConfigruation,这些自动配置类中，会根据一个叫configurationProperties的注解绑定一个类，获取所有自动配置的默认值，而这个类它又和yml文件进行了绑定，我们只需要改变yml的配置即可自定义自己的自动配置

Java | 复制代码

```java
//表示这是一个配置类，和以前编写的配置文件一样，也可以给容器中添加组件；
@Configuration

//启动指定类的ConfigurationProperties功能；
  //进入这个HttpProperties查看，将配置文件中对应的值和HttpProperties绑定起来；
  //并把HttpProperties加入到ioc容器中
@EnableConfigurationProperties({HttpProperties.class})

//Spring底层@Conditional注解
  //根据不同的条件判断，如果满足指定的条件，整个配置类里面的配置就会生效；
  //这里的意思就是判断当前应用是否是web应用，如果是，当前配置类生效
@ConditionalOnWebApplication(
    type = Type.SERVLET
)

//判断当前项目有没有这个类CharacterEncodingFilter；SpringMVC中进行乱码解决的过滤器；
@ConditionalOnClass({CharacterEncodingFilter.class})

//判断配置文件中是否存在某个配置：spring.http.encoding.enabled；
  //如果不存在，判断也是成立的
  //即使我们配置文件中不配置pring.http.encoding.enabled=true，也是默认生效的；
@ConditionalOnProperty(
    prefix = "spring.http.encoding",
    value = {"enabled"},
    matchIfMissing = true
)

public class HttpEncodingAutoConfiguration {
    //他已经和SpringBoot的配置文件映射了
    private final Encoding properties;
    //只有一个有参构造器的情况下，参数的值就会从容器中拿
    public HttpEncodingAutoConfiguration(HttpProperties properties) {
        this.properties = properties.getEncoding();
    }

    //给容器中添加一个组件，这个组件的某些值需要从properties中获取
    @Bean
    @ConditionalOnMissingBean //判断容器没有这个组件?
    public CharacterEncodingFilter characterEncodingFilter() {
        CharacterEncodingFilter filter = new OrderedCharacterEncodingFilter();
        filter.setEncoding(this.properties.getCharset().name());

        filter.setForceRequestEncoding(this.properties.shouldForce(org.springfra
```

```
mework.boot.autoconfigure.http.HttpProperties.Encoding.Type.REQUEST));
43
 filter.setForceResponseEncoding(this.properties.shouldForce(org.springfr
amework.boot.autoconfigure.http.HttpProperties.Encoding.Type.RESPONSE));
44          return filter;
45      }
46      //。。。。。。
47  }
```

@Conditional扩展注解	作用（判断是否满足当前指定条件）
@ConditionalOnJava	系统的java版本是否符合要求
@ConditionalOnBean	容器中存在指定Bean；
@ConditionalOnMissingBean	容器中不存在指定Bean；
@ConditionalOnExpression	满足SpEL表达式指定
@ConditionalOnClass	系统中有指定的类
@ConditionalOnMissingClass	系统中没有指定的类
@ConditionalOnSingleCandidate	容器中只有一个指定的Bean，或者这个Bean是首选Bean
@ConditionalOnProperty	系统中指定的属性是否有指定的值
@ConditionalOnResource	类路径下是否存在指定资源文件
@ConditionalOnWebApplication	当前是web环境
@ConditionalOnNotWebApplication	当前不是web环境
@ConditionalOnJndi	JNDI存在指定项

精髓

1、SpringBoot启动会加载大量的自动配置类 ① springboot ---启动加载自动配置类----面试官理论一番怎么得到的

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哪些组件；（只要我们要用的组件存在在其中，我们就不需要再手动配置了）

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。我们只需要在配置文件中指定这些属性的值即可；

② 自动配置类如何绑定yml文件的,我们手动覆盖自动配置的原理是什么

xxxxAutoConfigurartion：自动配置类；给容器中添加组件

xxxxProperties:封装配置文件中相关属性；

那么多的自动配置类，必须在一定的条件下才能生效；也就是说，我们加载了这么多的配置类，但不是所有的都生效了。

我们怎么知道哪些自动配置类生效?

我们可以通过启用 debug=true属性；来让控制台打印自动配置报告，这样我们就可以很方便的知道哪些自动配置类生效；

#开启springboot的调试类 debug=true

Positive matches:（自动配置类启用的：正匹配）

Negative matches:（没有启动，没有匹配成功的自动配置类：负匹配）

Unconditional classes: （没有条件的类）

9.如果在springboot遇到****configruation

这个配置类是用于扩展的，例如mvcConfigruation----我们可以保留springboot的默认设置，又可以对其实现进行扩展自己的配置。

10.templates的使用

HTML 复制代码

```
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<link th:href="@{css/default.css}" rel="stylesheet" type="text/css" />
<div th:text="">[[#{username配置文件中的}]]</div>
 th:可以修饰html中所有的标签
```