1.Spring Boot Maven项目的依赖配置文件是:pom.xml，默认的视图模板文件夹是: resources\_

2. 2.Spring Boot 项目的打包命令是:\_ package\_\_\_\_\_,两种打包发布方式分别是:jar、

war.

3.在 Spring Boot中，可以使用分页插件\_PageHelper\_\_来实现分页查询。

4.在 Spring Boot 中，可以使用自定义注解和\_\_切面类或者@Aspect\_来实现AOP。

5.Spring Boot 可以通过使用注解\_\_@RabbitListener \_和相关类(如Message)来实现消息队列的消费者。

6.在Spring Boot中，可以使用配置文件\_application.properties 或者application.yml\_\_来配置连接池的属性。

7.在 Spring Boot 中，可以使用注解\_\_@Transactional \_\_和事务管理器(如 JpaTransactionManager)来实现数据库事务管理。

8.在 Spring Boot 中，可以使用\_JUnit 和相关注解(如@SpringBootTest)来进行单元测试。

9.在 Spring Boot 中，可以使用\_\_@RestController \_注解和相关注解(如@GetMapping、@PostMapping)来实现RESTful API

10.Spring Boot 可以使用\_\_RestTemplate\_的方式来实现跨服务通信

二．选择

1.Spring boot 如何添加开发者工具集( A )

A:spring-boot-devtools

B: spring-boot-starter

C:spring-starter-ribbon

D:spring-starter-feign

2.下列关于 rabbitMQ描述错误的是( D )

A:rabbitMQ是基于消息发布/订阅模式实现的消息系统

B:高吞吐:在廉价的商用机器上也能支持单机每秒100K条以上的吞吐量

C:实时性:支持实时数据处理和离线数据处理

D:不支持水平扩展

3.下列关于 rabbitMQ中涉及的一些基本概念错误的是( D )

A:Topic:(主题)是特定类型的消息流。消息是字节的有效负载(Payload)，话题是消息的分类名或种子(Feed) 名。

B:Producer(生产者):是能够发布消息到话题的任何对象。

C:Broker(服务代理):已发布的消息保存在一组服务器中，它们被称为代理(Broker)或Kafka集群。

D:Consumer(消费者):只能订阅一个话题，并从Broker拉数据，从而消费这些已发布的消息。

4.spring事件机制是使用( A )

A:订阅和发布模式 、B:只有订阅模式

C:只有发布模式 D:都错

5.如何使用@FeignClient访问外网 http接口( A )

A:@FeignClient(name = "baidu",url = "http://192.168.0.194:8082")访问

B: @FeignClient(name = "baidu",uri= "http://192.168.0.194:8082")访问

C: @FeignClient(name = "baidu")访问

D:以上都不对

6.下列对Spring Cloud Config项目正确的是( A )

A:配置管理开发工具包，可以让你把配置放到远程服务器。

B:目前只支持git不支持Subversion

C:目前不支持本地存储。

D:以上叙述都正确。

7.下列关于 springClound 不正确的是:( D )

A:服务发现Netflix Eureka

B: 客服端负载均衡Netflix Ribbon

C:断路器Netflix Hystrix

D:分布式配置Netflix Zuul

8.springboot 默认读取配置文件是( A )

A．oapplication.properties 或application.ymt

B:applicationContext.xmt

C：ContextLoader.properties

D:applicationContextxml或ContextLoader.properties

9.下列关于 SpringBoot的优点错误的是:( B )

A:为所有Spring开发者更快的入门

B:外嵌式容器简化 Web 项目

C:开箱即用，提供各种默认配置来简化项目配置

D:没有冗余代码生成和XML配置的要求

10.下列关于SpringBoot热部署方式不正确的是:( D )

A、在 plugin标签中加入插件org.springframeworkspringloaded1.2.6.RELEASE进入pom 文件的目录输入mvn spring-boot:run启动

B、启动的加入参数引用springloaded jar 包 -javaagent:'jar包路径

-noverify

C、pom文件加入依赖 org-springframework.boot spring-boot-devtools

true

D: 都不是

ADDAA

ADABD

三，程序填空

springboot项目，做一个校验的功能，请按照提示补全下划线处的空白代码。

页面com.beans.UserInfo.java:

1. (message="账号不能为空")

private String userName;

(2)(min=5,max=10, message="密码长度必须在5-10之 private String password;

(3)\_ \_(message="{phone.not.empty}")

(4)\_\_(regexp ="I\\d{10}",message=" {phone.not.correct}")

private String phone;

@ (5) (message="' {(6)}”)//源文件中读信息读取邮件

private String email;

application.properties:

spring.datasource.username=root

spring.datasource.password=123456

Spring.datasource.driverClassName=(7)

spring.datasource.url=jdbe:mysql://ocalhost:3306/hebxysys?characterEncodin=ut8&useSSL-false&rewriteBatchedStatements=true&serverTimezone-UTC

spring.datasource.type=(8)

mybatis.type-aliases-package=(9)

spring.mvc.view.prefix=/WEB-INF/view/

spring.mvc.view.suffix=,jsp

spring.mvc,format,date=(10) //年月日就行

serverport-8081

1. @NotBlank

2. @Length

3. @NotEmpty

4. @Pattern

5. @Email

6.@email.not.correct

7. com.mysql.cj.jdbc.Driver

8. com.alibaba.druid.pool.DruidDataSource

9. com.beans

10.yyyy-MM-dd

四、程序阅读题(每题5分，共10分)

1. 说明下边标记注释的含义

<beans>

<context:component-scan base-package="com.controller"/> (1)

<mvc:resources location="/static/" mapping="/static/\*\*"></mvc:resources>

<mvc:default-servlet-handler/> (2)

<bean

class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="prefix" value="/WEB-INF/view/" /><property name="suffix" value=".jsp" >

</bean> (3)

<bean

class="org.springframework.web.servlet.handler.SimpleMappingExceptionResolve r"> (4)

<property name="exceptionMappings">(5)

<props>

<prop

key="com.beans.MyException">error\_proccess\_1</prop>

</props>

</property>

</bean>

</beans>

1)开启自动扫描,扫描本项目com.controlle 包

2) 解决静态文件问题

3) 视图解析器

4)全局异常配置

5)异常映射表

2.application 配置文件，标注说明含义

spring.datasource.username=root spring.datasource.password=123456

spring.datasource.driverClassName=com.mysql.cj.jdbc.Driver

spring.datasource.url (1)

spring.datasource.type (2)

mybatis.type-aliases-package (3)

spring.mvc.format.date(4)

server.port (5)

程序阅读第二题

1） 指定数据库连接的 URL地址（1分）

2） 数据库连接池类型（1分）

3） 包的别名（1分）

4） SpringMVC 格式化日期（1分）

5）服务端口号（1分）

六、程序设计题(每题10分，共40分)

1.在 SpringBoot 中，添加一个拦截器，除了/user/login不拦截之外全部拦截，当 url指向/eee的时候，拦的同时并指向视图abc.jsp，编写拦截器部分即可，请根据此要求完成控制器程序。(拦截器类名为MvcConfig)

(1)

public class MvcConfig implements WebMveConfigurer {

public void addViewControllers(ViewControllerRegistry registry){

Registry.addViewController("/eee").set ViewName("abc");

}

public void addInterceptors(InterceptorRegistry registry) {

String[] excludePaths = {"/user/login" };

Registry.addInterceptor(new SessionInterceptor())

.addPathPatterns("/\*\*”).excludePathPatterns(excludePaths);

}

@Bean（1分）

WebMveConfigurer webMveConfigurer() {

return this;

}

}

2.Springcloud 中,支付服务提供者 controller名为PaymentController ,service 名为 PaymentService，接口中有两个方法为addPayment(PaymentPay)-getPaymentByld(int id),封装的DTO 类名为R.class，枚举类为EnumStates,其中属性有OK(200，操作成功)和ERROR(-1,操作失败),还有value,message,有这俩参数的构造函数。请按照规则在controller中以restfull风格完成两个方法。

@PostMapping（“/addpayment”）（1分）

public R<?>add(@RequestBody Payment payment){

int result=paymentService.addPayment(payment);

if(resulte==1) {

return new R<>(EnumStatus.OK.getValue(), EnumStatus.OK.getMessageO,"生成

的自増主键是："+payment.getId());

}

else {

return new R<>(EnumStatus.ERROR.getValue(),

EnumStatus. ERROR.getMessage(),null);

}

}

@GetMapping(“/getpayment/{id}”)

public R<Payment> get(@Path Variable("id") int id) {

Payment payment=paymentService.getPaymentByid(id);

return new R<>(EnumStatus.OK.getValue(),

EnumStatus.OK.getMessage(),payment);

}

3.承接上题,Springcloud中服务消费者controller名为OrderController，需要定义支付服务地址常量名为PAYMENT\_URL,并且需要远端的调用服务地址(端口是10001)，模板发送类为RestTemplate,DTO 类名为R，请在OrderController中写出调用支付服务端接口的两个方法，接口名为: addPayment(Payment pay)和 getPaymentById(int id).

答:

private final static String PAYMENT\_URL="http://localhost:10001";

@GetMapping("/getpayment/{id}")

public R<?> getPayment(@PathVariable("id") int id){

return testTemplate.getForObject(PAYMENT\_URL+"/payment/getpayment"+id, R.class);

}

@PostMapping("/addpayment")

public R<?>addpayment(Payment payment){

R<?>r=

testTemplate.postForObject(PAYMENT\_URL+"/payment/addpayment",payment,R.class);

return r;

}

4.编写 dockerfile 文件

FROM java:8

COPY \*.jar/app-jar

CMD ["-server.port=8081”]

EXPOSE 8081

ENTRYPOINT ["java","-jar"."/app.jar"]