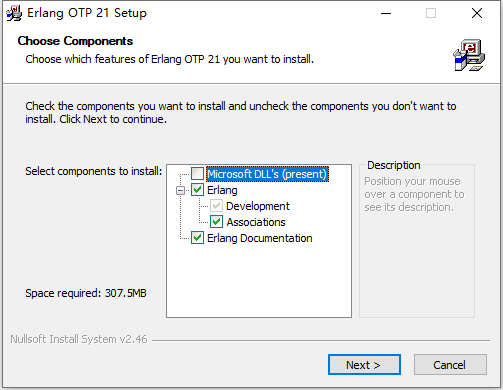
# 实验十七 SpringBoot整合RabbitMQ

一、安装Erlang语言包

1.安装

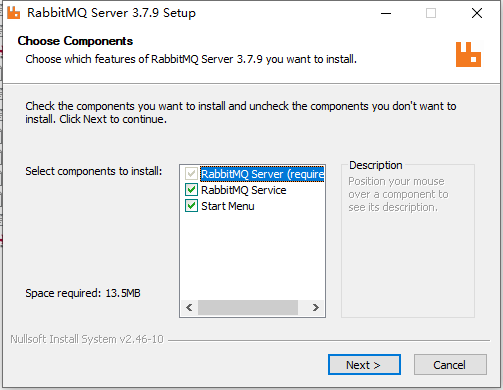


2.自动配置环境变量

C:\Program Files\erl10.2\bin 放入到path环境变量中

二、安装RabbitMQ

1.安装



将

C:\Program Files\RabbitMQ Server\rabbitmq\_server-3.7.9\sbin

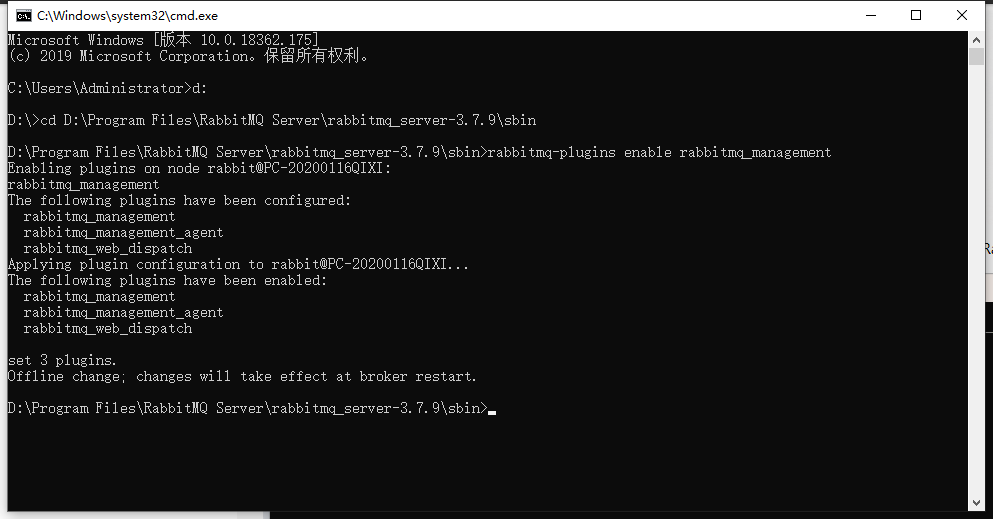
放入到path环境变量中

2.启动服务

（1）安装插件并启动服务

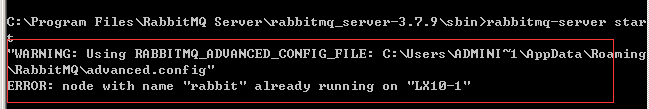
安装管理器插件命令：rabbitmq-plugins enable rabbitmq\_management

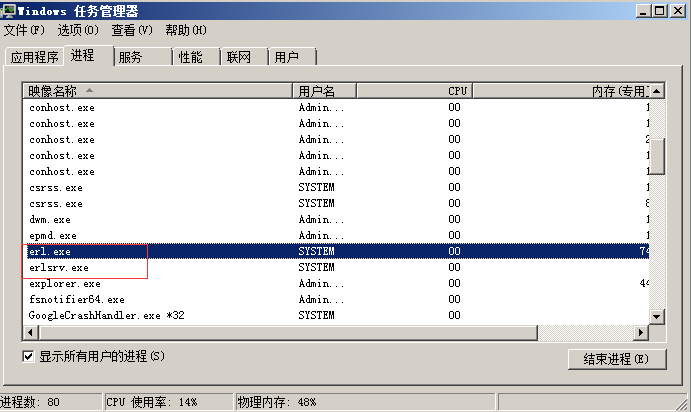
启动服务命令：rabbitmq-server start





如果出现下图错误，须将语言服务关闭。





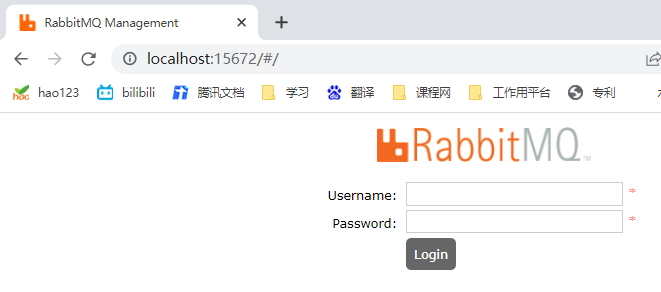
使用cmd窗口的dos命令杀死进程可使用如下命名：

taskkill /IM erl.exe /F

taskkill /IM erlsrv.exe /F

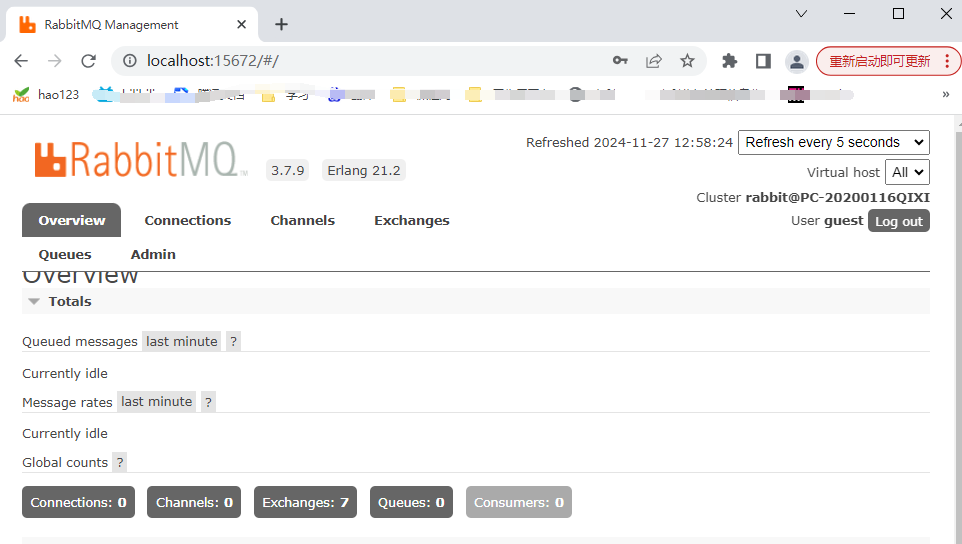
（2）使用管理器

在地址栏中录入：<http://localhost:15672>



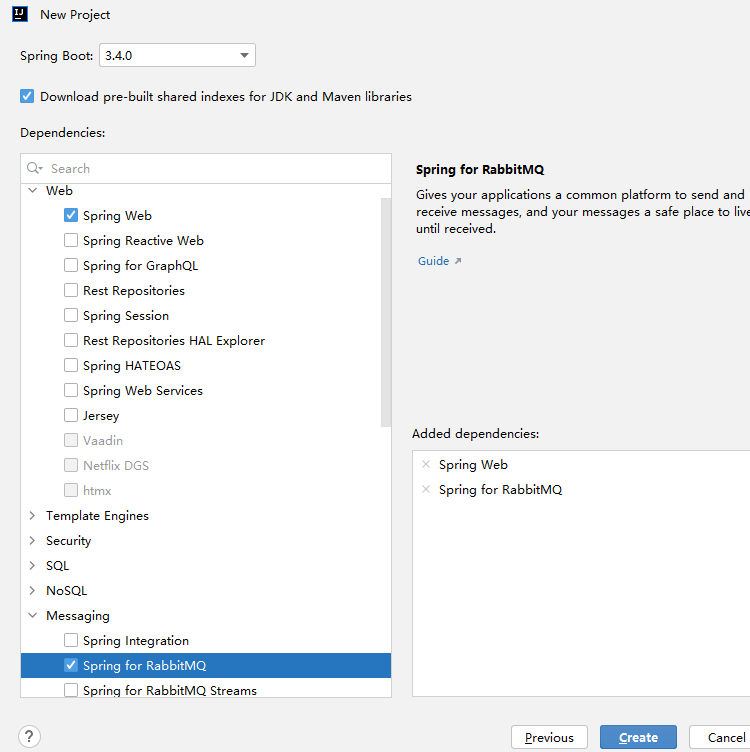
录入默认的用户名和密码 : guest/guest





三、SpringBoot整合RabbitMQ

1.使用Spring Initializr方式创建项目，选择spring web 依赖和Spring for RabbitMQ依赖



2.修改Pom.xml 和设置本地仓库

3.修改全局配置文件

spring.rabbitmq.host=localhost  
spring.rabbitmq.port=5672  
spring.rabbitmq.username=guest  
spring.rabbitmq.password=guest  
spring.rabbitmq.virtual-host=/

四、Publish/Subscribe模式（有三种方式）

1.API方式

（1）使用AmqpAdmin定制消息发送组件

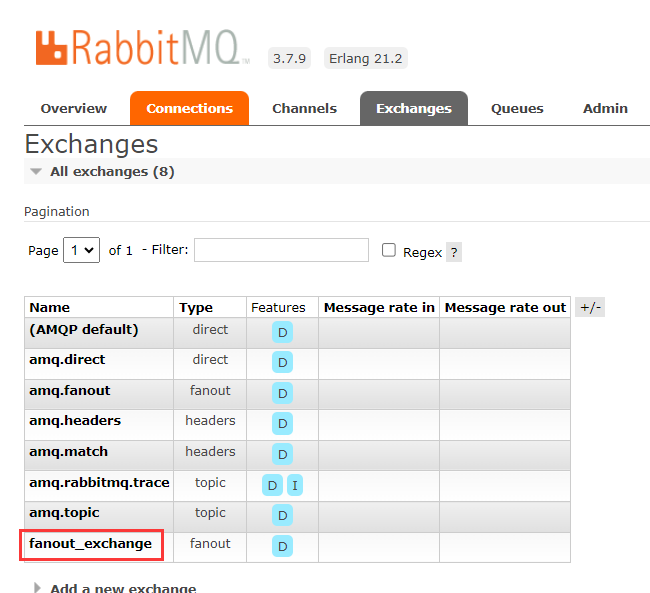
在测试类Chap08ApplicationTests中注入实例并添加测试方法



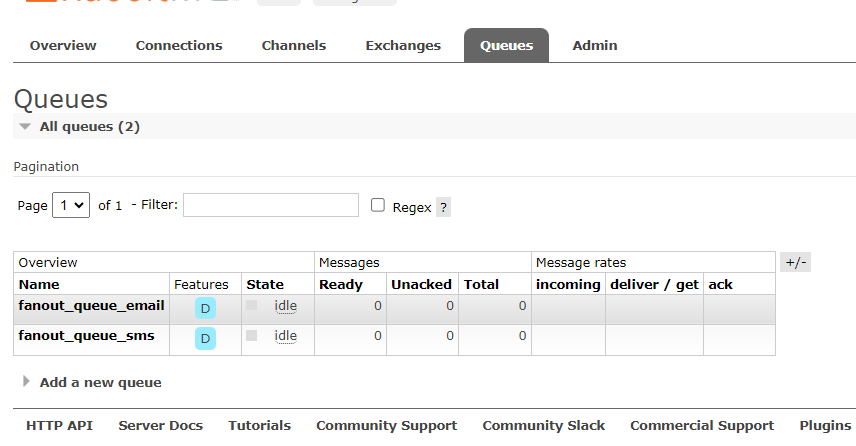
@SpringBootTest  
public class Chap08ApplicationTests {  
 @Autowired  
 private AmqpAdmin amqpAdmin;  
 @Autowired  
 private RabbitTemplate rabbitTemplate;  
  
 */\*\*  
 \* 使用AmqpAdmin管理员API定制消息组件  
 \*/* @Test  
 public void amqpAdmin() {  
 *// 1、定义fanout类型的交换器* amqpAdmin.declareExchange(new FanoutExchange("fanout\_exchange"));  
 *// 2、定义两个默认持久化队列，分别处理email和sms* amqpAdmin.declareQueue(new Queue("fanout\_queue\_email"));  
 amqpAdmin.declareQueue(new Queue("fanout\_queue\_sms"));  
 *// 3、将队列分别与交换器进行绑定* amqpAdmin.declareBinding(new Binding("fanout\_queue\_email",Binding.DestinationType.*QUEUE*,"fanout\_exchange","",null));  
 amqpAdmin.declareBinding(new Binding("fanout\_queue\_sms",Binding.DestinationType.*QUEUE*,"fanout\_exchange","",null));  
 }

}

（2）测试







（3）创建User类

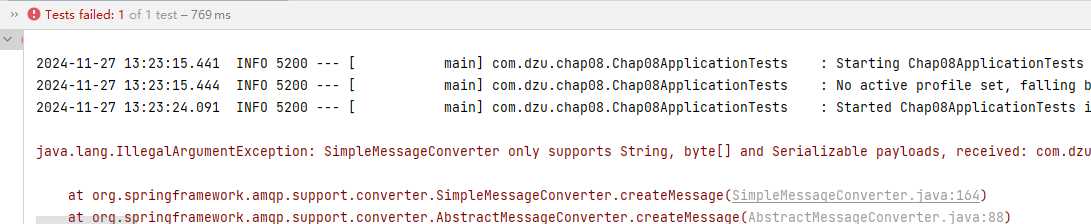
public class User {  
 private Integer id;  
 private String username;  
  
 public Integer getId() {  
 return id;  
 }  
  
 public void setId(Integer id) {  
 this.id = id;  
 }  
  
 public String getUsername() {  
 return username;  
 }  
  
 public void setUsername(String username) {  
 this.username = username;  
 }  
  
 @Override  
 public String toString() {  
 return "User{" +  
 "id=" + id +  
 ", username='" + username + '\'' +  
 '}';  
 }  
}

（4）修改测试文件



@Test  
public void psubPublisher() {  
 User user=new User();  
 user.setId(1);  
 user.setUsername("石头");  
 rabbitTemplate.convertAndSend("fanout\_exchange","",user);  
}

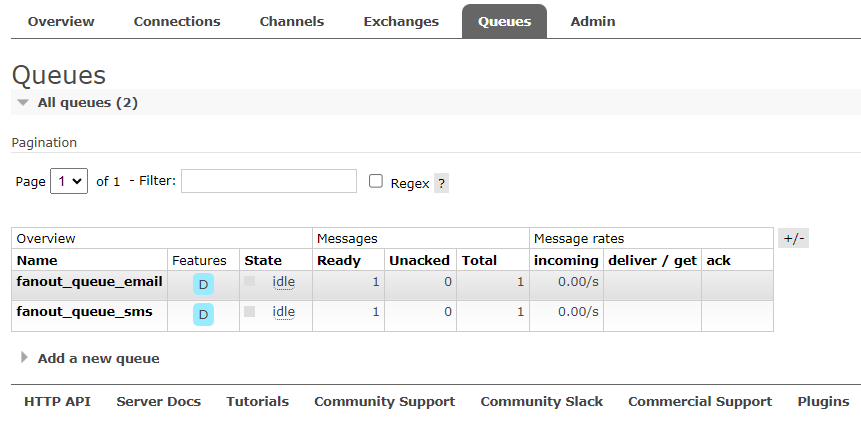
（5）运行测试文件



（6）创建RabbitMQConfig类

@Configuration  
public class RabbitMQConfig {  
 */\*\*  
 \* 定制JSON格式的消息转换器  
 \* @return  
 \*/* @Bean  
 public MessageConverter messageConverter(){  
 return new Jackson2JsonMessageConverter();  
 }  
  
 }

（7）测试



（8）消息消费者接收消息

创建RabbitMQService类



package com.dzu.chap08.service;  
  
import com.dzu.chap08.domain.User;  
import org.springframework.amqp.core.Message;  
import org.springframework.amqp.rabbit.annotation.Exchange;  
import org.springframework.amqp.rabbit.annotation.Queue;  
import org.springframework.amqp.rabbit.annotation.QueueBinding;  
import org.springframework.amqp.rabbit.annotation.RabbitListener;  
import org.springframework.stereotype.Service;  
  
  
@Service  
public class RabbitMQService {

@RabbitListener(queues = "fanout\_queue\_email")  
public void psubConsumerEmail(Message message) {  
 byte[] body = message.getBody();  
 String s = new String(body);  
 System.*out*.println("邮件业务接收到消息： "+s);  
  
}  
  
@RabbitListener(queues = "fanout\_queue\_sms")  
public void psubConsumerSms(Message message) {  
 byte[] body = message.getBody();  
 String s = new String(body);  
 System.*out*.println("短信业务接收到消息： "+s);  
}

}

运行启动类

