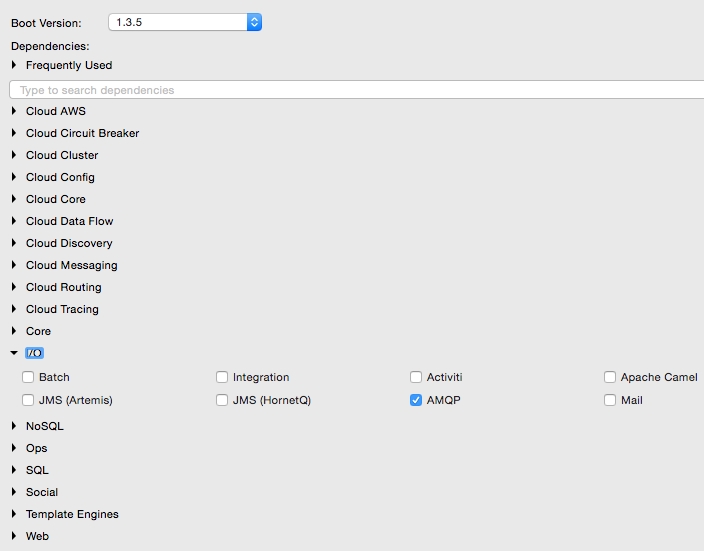
* Lab 8: Implementing Spring Boot Messaging
* Create a new project using STS to demonstrate this capability. In this example, instead of selecting **Web**, select **AMQP** under **I/O**:
* 
* Rabbit MQ will also be needed for this example. Download and install the latest version of Rabbit MQ from <https://www.rabbitmq.com/download.html>.
* Rabbit MQ 3.5.6 is used in this book.
* Follow the installation steps documented on the site. Once ready, start the RabbitMQ server via the following command:
* **$./rabbitmq-server**
* Make the configuration changes to the application.properties file to reflect the RabbitMQ configuration. The following configuration uses the default port, username, and password of RabbitMQ:
* spring.rabbitmq.host=localhost  
  spring.rabbitmq.port=5672  
  spring.rabbitmq.username=guest  
  spring.rabbitmq.password=guest
* Add a message sender component and a queue named TestQ of the org.springframework.amqp.core.Queue type to the Application.java file under src/main/java. RabbitMessagingTemplate is a convenient way to send messages, which will abstract all the messaging semantics. Spring Boot provides all boilerplate configurations to send messages:
* @Component   
  class Sender {  
   @Autowired  
   RabbitMessagingTemplate template;  
   @Bean  
   Queue queue() {  
   return new Queue("TestQ", false);  
   }  
   public void send(String message){  
   template.convertAndSend("TestQ", message);  
   }  
  }
* To receive the message, all that needs to be used is a @RabbitListener annotation. Spring Boot autoconfigures all the required boilerplate configurations:
* @Component  
  class Receiver {  
   @RabbitListener(queues = "TestQ")  
   public void processMessage(String content) {  
   System.out.println(content);  
   }  
  }
* The last piece of this exercise is to wire the sender to our main application and implement the runmethod of CommandLineRunner to initiate the message sending. When the application is initialized, it invokes the run method of CommandLineRunner, as follows:
* @SpringBootApplication  
  public class Application implements CommandLineRunner{  
    
   @Autowired  
   Sender sender;  
     
   public static void main(String[] args) {  
   SpringApplication.run(Application.class, args);  
   }  
     
   @Override  
   public void run(String... args) throws Exception {  
   sender.send("Hello Messaging..!!!");  
   }  
  }
* Run the application as a Spring Boot application and verify the output. The following message will be printed in the console:
* **Hello Messaging..!!!**