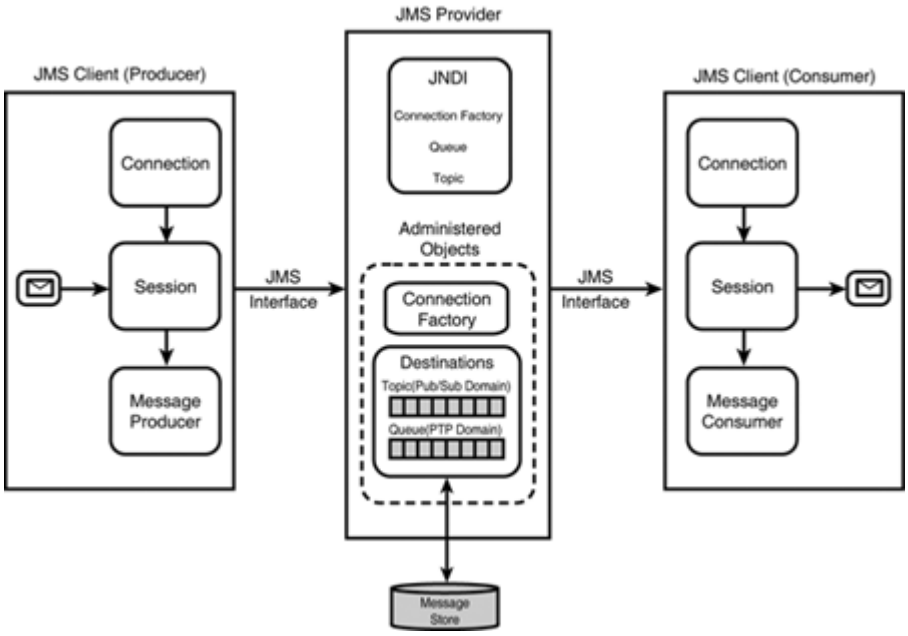
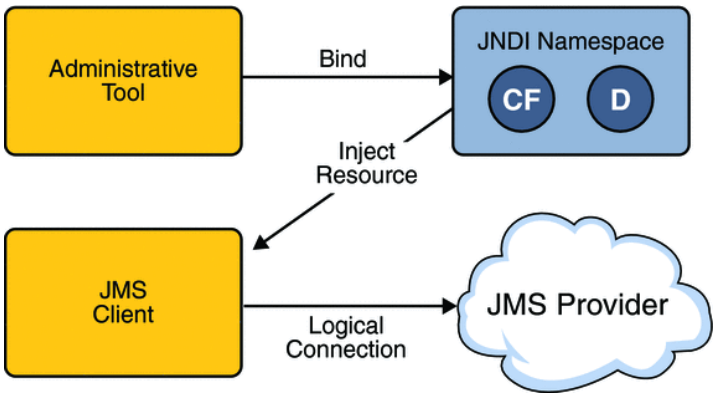
Билет ?:

1. Распределённая обработка в JMS (Jakarta Messaging)

- Спецификация API для обмена сообщениями в составе платформы Jakarta/Java EE.
- Широко используется при асинхронном выполнении задач.
- Позволяет инкапсулировать инфраструктурное ПО и протоколы обмена сообщениями за универсальным API.





Отправка сообщения:

- Лукапом из контекста мы получаем ConnectionFactory, с помощью которой создаём соединение Connection, с помощью которого создаём сессию Session.
- Лукапом из контекста вытаскиваем очередь по её имени (объект Destination).
- Создаём из полученной сессии producer, привязанный к очереди и отправляем сообщение: producer.send(message);

Получение сообщения:

- Лукапом из контекста мы получаем ConnectionFactory, с помощью которой создаём соединение Connection, с помощью которого создаём сессию Session.
- Лукапом из контекста вытаскиваем очередь по её имени (объект Destination).

- Создаём из полученной сессии consumer, привязанный к очереди. Создаем объект MessageListener и сетим его консьюмеру: consumer.setMessageListener(listener);

2. Bpms описание, плюсы, минусы

BPMS:

- Приложения, автоматизирующие управление бизнес-процессами предприятия.
- Позволяют задавать правила управления бизнес-процессом в виде диаграмм (например, BPMN).
- Могут быть отдельными приложениями, могут – библиотеками, встраиваемыми в ИС.
- Могут управлять только бизнес-логикой, могут ещё и генерировать интерфейс.

Плюсы и минусы:

- [+] Гибкая адаптация бизнес-процессов под меняющиеся требования.
- [+] Возможна модификация бизнес-процесса без участия специалиста (теоретически).
- [+] Автоматизируемый процесс должен быть специфицирован.
- [+*] Процессы не просто документируются, но и выполняются. Это позволяет предварительно протестировать их и откорректировать слабые места перед тем, как интегрировать процесс в повседневную работу
- [−] Сгенерированный интерфейс обычно неудобен.
- [−] Генерируемая программно логика может быть неэффективной в плане быстродействия.
- [−] “Внедрение ради внедрения”.

3. Написать планировщик, используя Java SE, который запускается каждый день в 00.00 и отправляет сообщения от студентов преподавателю, когда тот проверит рубежку. Во избежание ddos атаки интервал между сообщениями 5 секунд.

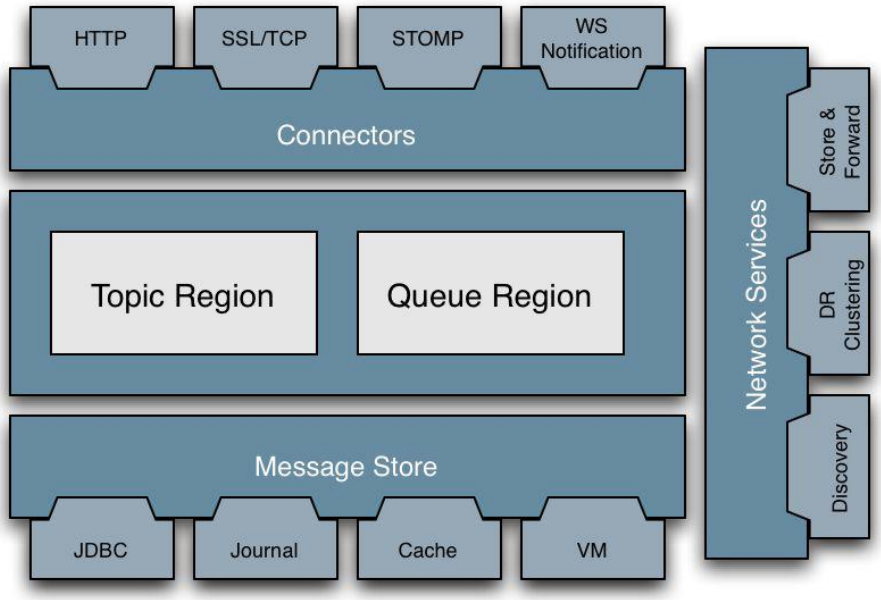
```
class PingTsopa {
    private final ScheduledExecutorService scheduler =
       Executors.newScheduledThreadPool(1);

    public void pingScheduledTask() {
        final Runnable ping = new Runnable() {
            ScheduledFuture<?> taskHandle = msgScheduler.schedule(new Runnable() {
                public void run() {
                    sendSpamToTsopa();
                }
            }, 5, TimeUnit.SECONDS);
        };

        long midnight = LocalDateTime.now()
            .until(LocalDate.now().plusDays(1).atStartOfDay(), ChronoUnit.MINUTES);

        scheduler.scheduleAtFixedRate(ping, //runnable
                                      midnight - System.currentTimeMillis(), // initialDelay
                                      TimeUnit.DAYS.toMillis(1) //period
                                      TimeUnit.MILISECONDS //unit
                                      );
    }
 }
```

Билет ?:

1. Active mq, архитектура, +/-, область применения

- Брокер сообщений от Apache.
- OpenSource (ASL), написан на Java.
- Полностью совместим со спецификацией JMS
- “Родной” протокол – OpenWire.
- Основные поддерживаемые протоколы – AMQP, STOMP и MQTT.

Архитектура:



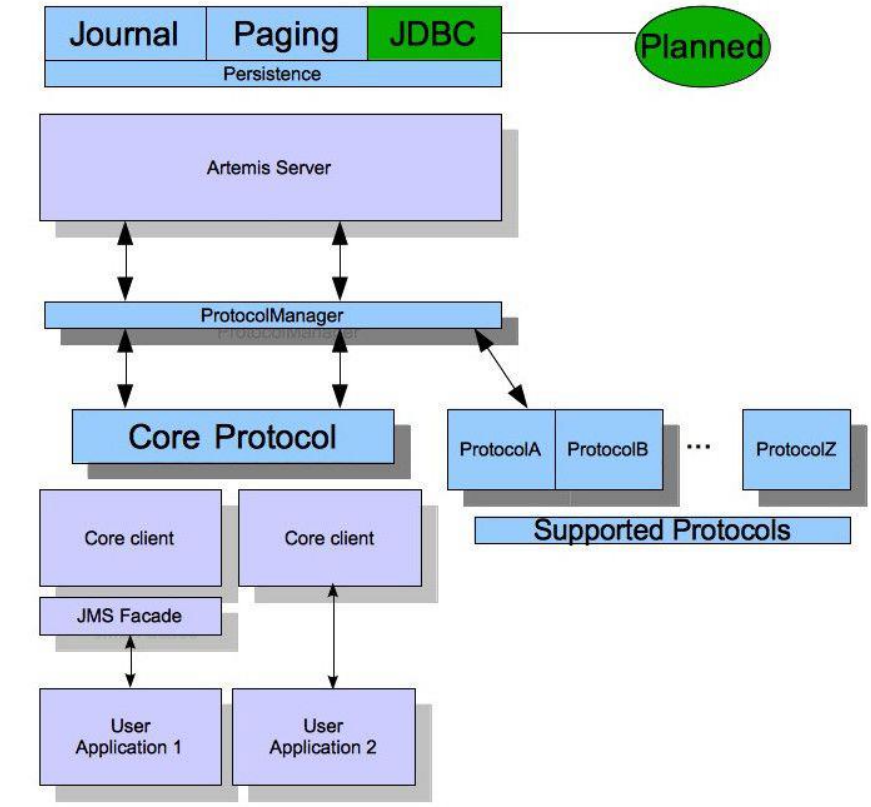


Figure 3.1 Artemis High Level Architecture

Область применения:

ActiveMQ используется в качестве моста связи между несколькими компонентами, которые могут быть размещены на отдельных серверах или могут быть записаны на разных языках программирования. Брокера сообщений можно встретить в корпоративных системах - или любых системах, которые имеют сложную архитектуру, например в финансовой и банковской промышленности, где важна надежная система с высокой доступностью.

Плюсы:

a) Высокая доступность, основанная на архитектуре ведущий-ведомый.

b) Надежность(низкая вероятность потери данных)

Минусы:

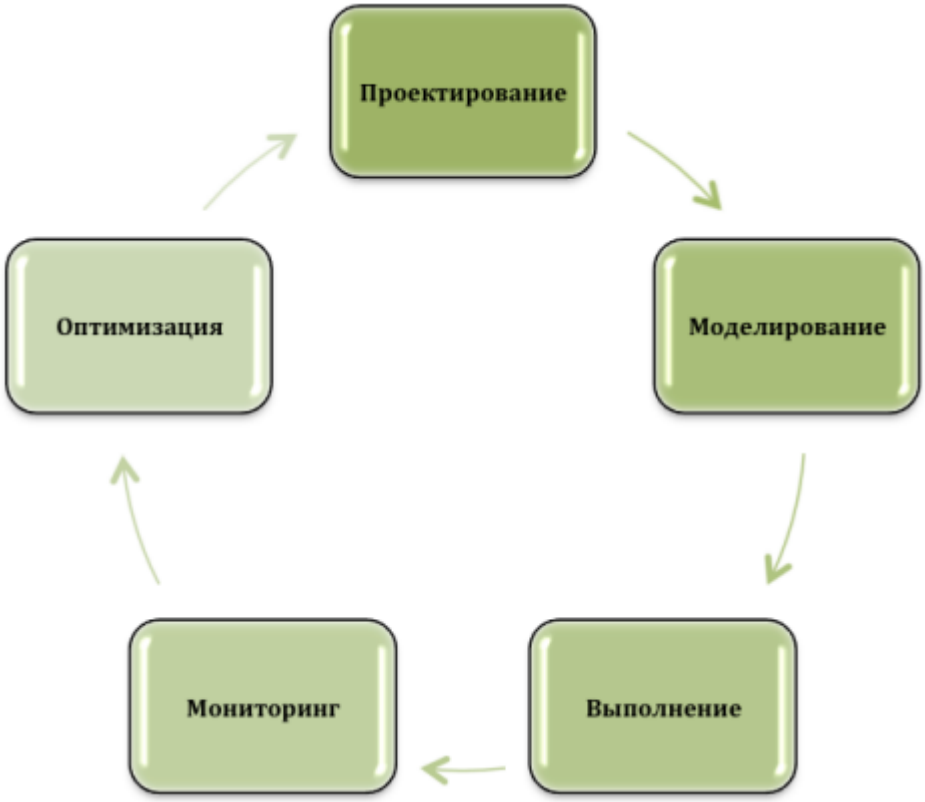
a). Пропускная способность на порядок ниже, чем у Kafka

b). Более тяжеловесный, чем RabbitMQ

2. Жизненный цикл BPM: фазы и что на них происходит

BPM — это концепция процессного управления организацией, сочетающая в себе идеологию и программное обеспечение для управления бизнес-процессами.

Жизненный цикл процесса – последовательность стадий и фаз, определяющих динамику реализации и развития процесса



- Проектирование – определение как бизнес-процесс существует на данный момент и понять, какими мы хотим его видеть в дальнейшем.
- Моделирование – на основании ранее составленного проекта создаём модель.
- Выполнение – запуск процесса в работу.
- Мониторинг – наблюдение и отслеживание процессов.
- Оптимизация – анализируете данные, полученные в результате мониторинга, проектирования и моделирования.

3. С помощью jakarta ee сделать планировщик: Блокировать стулом дверь аудитории после 5 минут от начала пары, если лектор в аудитории и включил проектор и при этом в аудитории есть пара. Получить расписание можно по isu api.

```
@Singleton
public class ChairSetterBean implements ChairSetter {
    @Inject
    private AudienceService audienceService;
    @Inject
    private ISUapi isuAPI;

    @Schedules ({
        @Schedule(dayOfWeek="1, 2, 3, 4, 5, 6", hour="8", minute="25"),
        @Schedule(dayOfWeek="1, 2, 3, 4, 5, 6", hour="10", minute="5"),
        @Schedule(dayOfWeek="1, 2, 3, 4, 5, 6", hour="11", minute="45")
        //others schedules
    })
    public void setChair() {
        if (!audienceService.isProfessorInAudience() || !audienceService.isProjectorOn() ||
!isuAPI.isLessonInAudienceNow())
            return;

        audienceService.setChairInAuditory();
    }
}
```

Билет ?:

1. Протоколы их плюсы и минусы

Message Queue Telemetry Transport (MQTT):

- Сетевой протокол *прикладного* уровня.
- *Двоичный* протокол, работает поверх TCP/IP.
- Использует модель “издатель-подписчик”.
- Изначально предназначен для организации взаимодействия между устройствами.
- *Легковесный* и *простой* в использовании.
- Широко распространён в мире IoT.

Плюсы и минусы:

- [+] Легковесный, быстрый.
- [+] Нетребователен к пропускной способности сети.
- [+] Реализует несколько моделей доставки сообщений.
- [−] Негибкий: к примеру, поддерживается всего 5 видов сообщений об ошибке и нет возможности сообщить клиенту об ошибке после установления HTTP-соединения.

- [−] Могут быть проблемы с масштабируемостью.

Advanced Message Queuing Protocol (AMQP)

- *Двоичный, прикладного* уровня, обычно функционирует “поверх” TCP.
- Ключевой элемент инфраструктуры – AMQP-брокер.
- Основной элемент – *фрейм*.
- Сообщения описывают сами себя (*self-described*).
- Содержимое фрейма бывает 9 видов – open (connection), begin (session), attach (link), transfer (message), flow, disposition, detach (link), end (session), close (connection).

Плюсы и минусы:

- [+] Мощнее, чем MQTT – больше видов сообщений, больше возможностей для их обработки.
- [+] Поддерживает транзакции.
- [+] Поддерживает различные механизмы оповещения об обработке сообщений.
- [−] “Тяжелее”, чем MQTT.
- [−] Небогатый выбор библиотек и API.

Streaming Text Oriented Messaging Protocol (STOMP)

- *Текстовый* протокол передачи сообщений.
- *Прикладного* уровня, похож на HTTP.
- Основная “философия” – *простота* и *читаемость сообщений*, отсутствие в них лишней метаинформации и разметки.

Плюсы и минусы:

- [+] Текстовый.
- [+] Простой.
- [+] Хорошо читаемый.
- [−] Текстовый.
- [−] Простой.

eXtensible Messaging and Presence Protocol (XMPP):

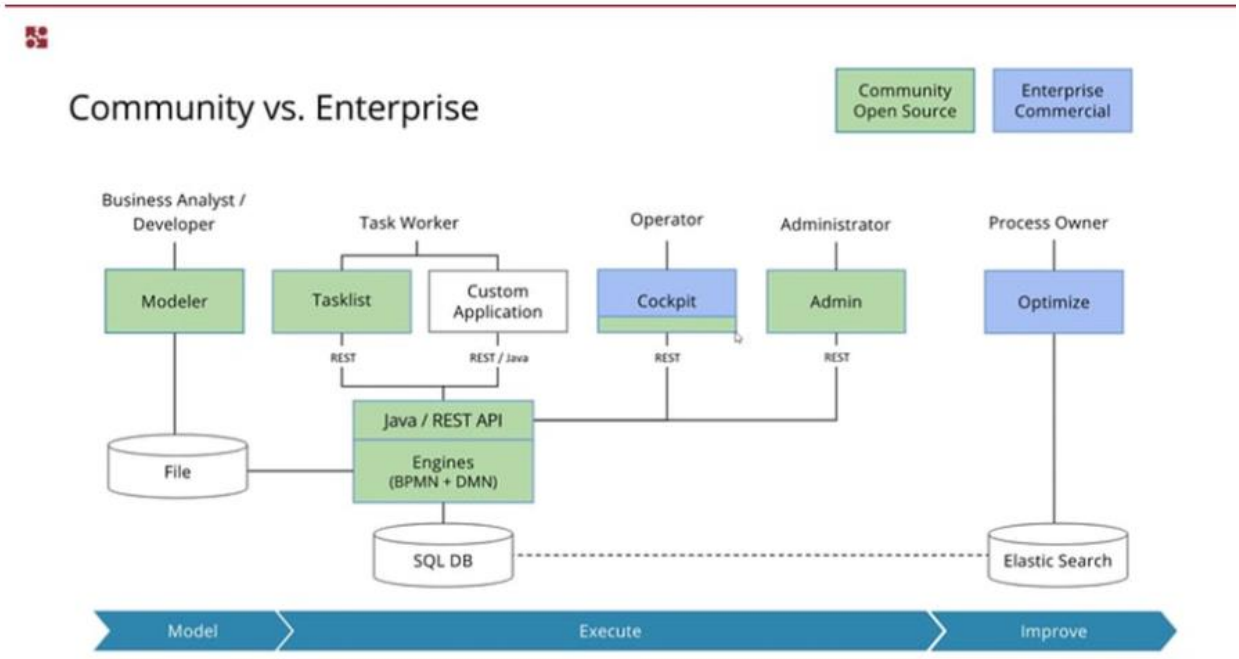
- *Прикладного* уровня, может работать “поверх” TCP или HTTP.
- *Текстовый, основан на XML.*
- Изначально спроектирован *расширяемым* – поддерживает передачу голоса, видео, файлов и т. д.
- Для *интеграции* с другими протоколами есть механизм транспортов (*transports*).

Плюсы и минусы:

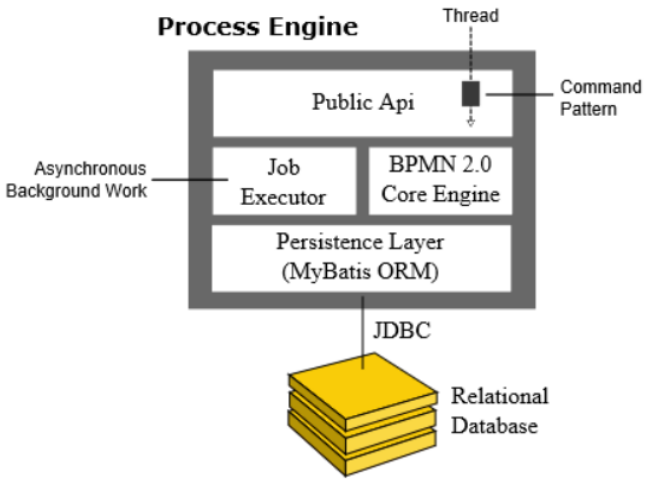
- [+] Гибкий.
- [+] Мощный.
- [+] Расширяемый.
- [+] Текстовый.
- [+] Популярный, хорошая поддержка на уровне библиотек.
- [−] Текстовый.
- [−] У XML читаемость хуже, чем у plain text.

- [−] Достаточно “тяжеловесный”.
- [−] Предназначен для IM – “забивание гвоздей микроскопом”.

2. Основные компоненты ~~jbms~~ (Camunda?)



- **Modeler** — это приложение для создания моделей BPMN процессов. Эти модели нужны для других частей системы.
- **Task list** — это веб-приложение, в котором исполнители выполняют задачи, поставленные на них бизнес-процессом.
- **BPMN Engine —** это непосредственно движок, которые отвечает за интерпретацию BPMN в объекты JAVA, сохранение объектов в базе и реализацию других вещей (типа листенеров активностей), которые крутятся вокруг процессов.
- **DMN Engine** — аналогично BPMN Engine, только для DMN.



- **Cockpit** — это веб-приложение для просмотра состояния процессов. В бесплатной версии он сильно обрезан по функционалу.

- **Admin —** это веб-приложение для управления правами пользователей и пользователями.
- **Optimize** — это веб-приложение для анализа бизнес-процессов. Оно платное.

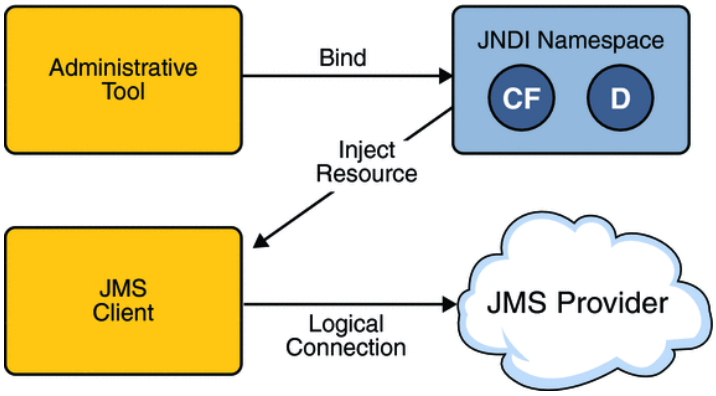
3. Написать Quartz, который каждый день в 10 и 22 проверяет у пациентов температуру, если было уже в 2 раза превышение, то вызвать врачей для проведения теста ПЦР, если тест положительный, то ограничить пациента и всех, кто с ним контактировал на 10 дней, все методы для работы с врачами уже есть

```
public class CovidJob implements Job {
    @Autowired
    PatientService pService;
    @Autowired
    DoctorService dService;

    public void execute(JobExecutionContext arg0) throws JobExecutionException {
        List<Patient> patients = pService.getAllPatients();
        for (Pacient p : pacients) {
            if (p.getTemperatureIncresement() > 2) {
                bool res = dService.makePCR(p);
                if (res = true) {
                    dService.imprison(p, 10);
                    dService.imprison(p.getContactedPatients(), 10);
                }
            }
        }
    }
}

public class Main {

    public static void main(String args[]) {
        SchedulerFactory schedFact = new StdSchedulerFactory();
        try {
            Scheduler sched = schedFact.getScheduler();
            JobDetail covidJob = JobBuilder.newJob(CovidJob.class)
                                        .withIdentity("CovidJob", "group1")
                                        .build();
            Trigger covidTrigger = TriggerBuilder.newTrigger()
                                        .withIdentity("covidTrigger", "group1")
                                        .withSchedule(CronScheduleBuilder.cronSchedule("0 10,22 *
* *"))
                                        .build();
            sched.scheduleJob(covidJob, covidTrigger);
            sched.start();

        } catch (ScheduledException e) {
            e.printStackTrace();
        }
    }
}
```

Билет 3:

1. Ресурсы и сообщения JMS

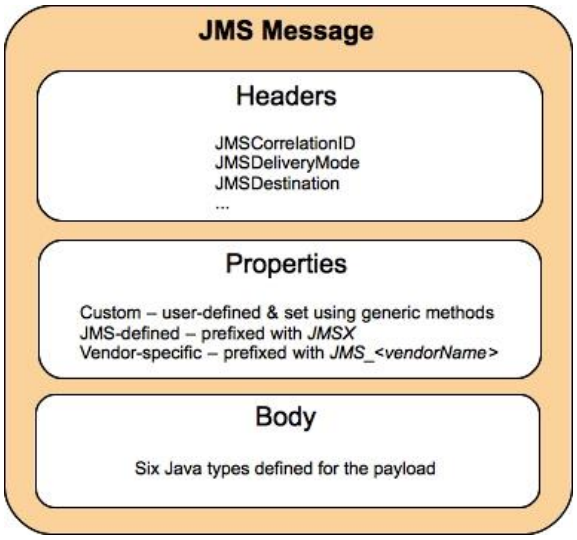
Любой компонент, использующий JMS, прежде всего должен создать соединение с JMS провайдером. Для этого, нужно создать необходимые ресурсы. Группа ресурсов JMS – это модуль адаптера ресурсов, обеспечивающий связывание фабрик соединений, очередей и т. п., которые мы можем получить с помощью JNDI lookup.



JMS состоит из определенных сообщений и клиентов обменивающимися ими.

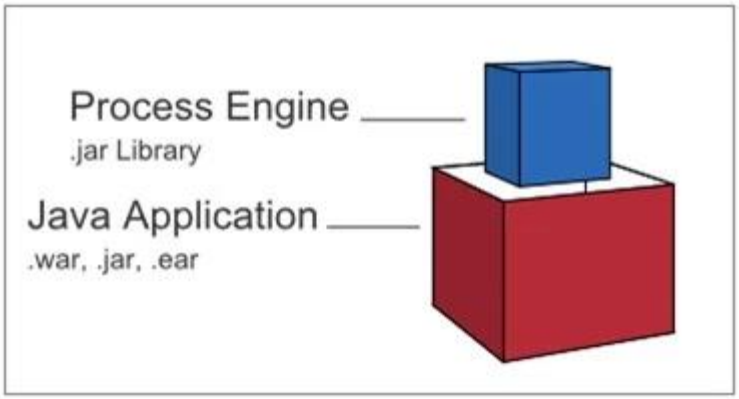
Сообщение состоит из трех частей: заголовок, свойств и тело.

- Заголовок – содержит информацию, которая используется для маршрутизации и идентификации сообщений (обязательный). Заголовок хранит мета информацию сообщения, заполняемую автоматически.
- Свойство – содержит информацию, которую можно использовать для фильтрации сообщений (время, дата и т. п.). Поле свойств схоже с заголовком, но оно заполняется программно, и позже получатель сможет прочитать эту информацию.
- Тело – содержит информацию о типе сообщения и полезную нагрузку. Тип нагрузки определяется при создании сообщения.

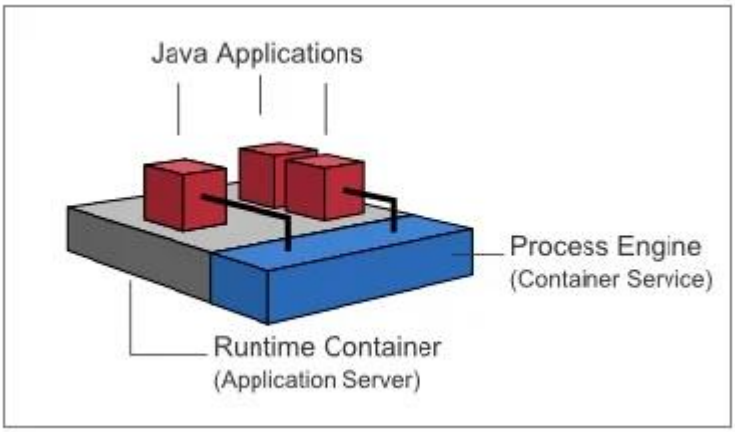


2. Установка и конфигурация ~~jbpm~~ (Camunda?)

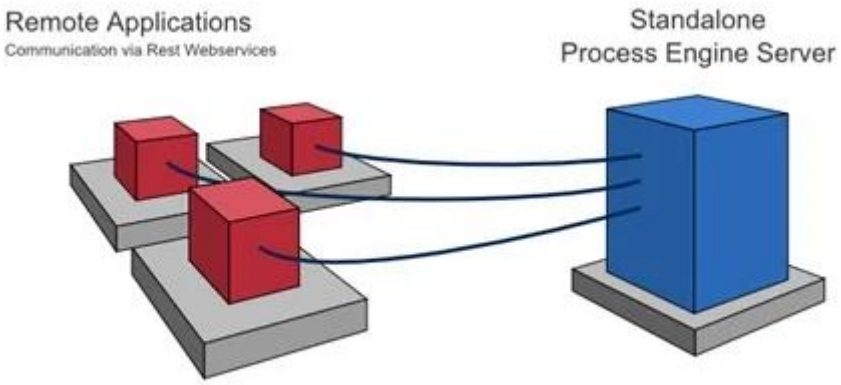
1 способ: встроенная библиотека в приложении.



2 способ: сервис внутри сервера приложений или контейнера сервлетов («расшаренный» сервис), который могут использовать другие приложения, развернутые в контейнере.



3 способ: standalone-сервер – развернуть Camunda как отдельно стоящий сервер и приложение будут с ним контактировать по Rest API удаленно.



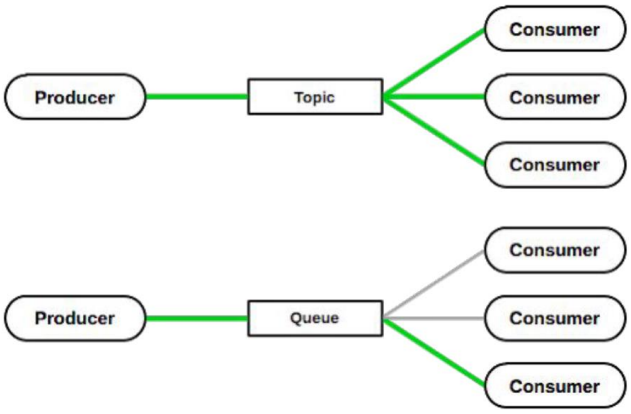
3. Планировщик на Spring: просыпается каждые 5 минут рабочего времени с 11:00 до 17:00 (по будням), исключая обеденные перерыв с 13:00 до 15:00 и выполняет:

- List<Student> checkOutWear();
- List<Place> checkChairAngles(5);
- List<Student> checkFood(); List<Student> checkDrinks();
- Студентам, для которых выполняются хотя бы 2 условия из вышеперечисленных – printLetter

```
@EnableScheduling
public class RobotScheduledTask {

    //At every 5th minute from 0 through 55 past
    //every hour from 11 through 13 and every hour from 15 through 17
    //on every day-of-week from Monday through Friday
    @Scheduled(cron = "0-55/5 11-13,15-17 * * 1-5")
    public void scheduleTask() {
        List<Student> outweared = checkOutwear();
        List<Chair> chairsWithWrongAngle = checkChairAngles();
        List<Student> studentInWrongChairs = chairsWithWrongAngle
                        .stream()
                        .map(c -> c.getStudent())
                        .collect(Collectors.toList());
        List<Student> studentsWithFood = checkFood();
        List<Student> studentsWithDrinks = checkDrinks();
        List<Student> students = getAllStudentsInAudience();
        for (Student s : students) {
            int fuckups = 0;
            if (outweared.constins(s)) fuckups++;
            if (studentInWrongChairs.constins(s)) fuckups++;
            if (studentsWithFood.constins(s)) fuckups++;
            if (studentsWithDrinks.constins(s)) fuckups++;
            if (fuckups >= 2) printLetter(s);
        }
    }
}
```

Билет 4

1. Модели поставки сообщений в JMS



Точка-точка — это метод связи, при котором производители JMS отправляют сообщения в ***очередь***, а потребители JMS получают сообщения из этой очереди. Очередь работает по принципу FIFO (первый вошел, первый вышел), сообщения в очереди упорядочены на основе порядка, в котором они попали в эту очередь. Несколько производителей могут отправлять сообщения в очередь, а также несколько потребители могут получить доступ к этой очереди. Однако любое сообщение на этой очереди может быть получено только одним потребителем. *Данный метод обычно используется, когда нужно чтобы сообщение отправлялось только одному потребителю(приложению).*

Издатель-подписчик — это метод связи, при котором подписчики могут получать информацию, в виде сообщений, от издателей. Такие системы могут иметь несколько издателей и подписчиков. Издатели генерирует сообщения, называемые публикациями, а также определяют ***топик*** для этих сообщений. Подписчики в свою очередь создают подписки, которые описывают топик, в котором они заинтересованы. Подписчики могут сделать несколько подписок и получать сообщения от нескольких издателей. *Данный метод обычно используется, когда нужно распределить сообщения по всем потребителям, которые подписаны на определённый топик.*

2. Архитектура ~~jBPM~~ (Camunda)

См. вопрос о компонентах Camunda.

3. Написать планировщик в cron, который каждый день в 5 утра кроме января, июля и августа будет запускать три скрипта

```
0 5 * 2,3,4,5,6,9,10,11,12 *
./kill_all_student_processes.sh;./recursively_remove_all_large_files.sh;
./send_email_complaints.sh
```