НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерных технологий

Сервисно-ориентированная архитектура

Лабораторная работа № 3

Вариант 1006.69.12

Выполнил
студент

Демичев Даниил Дмитриевич

Группа № P34112

Преподаватель: Усков Иван Владимирович

г. Санкт-Петербург

2022

Задание

Переработать веб-сервисы из лабораторной работы #2 таким образом, чтобы они реализовывали основные концепции микросервисной архитектуры. Для этого внести в оба сервиса -- "вызываемый" (из лабораторной работы #1) и "вызывающий" (добавленный в лабораторной работе #2) перечисленные ниже изменения.

Изменения в "вызываемом" сервисе:

- Разделить приложение на два модуля -- веб-приложение с веб-сервисом и EJB-jar с бизнес-компонентами.
- Переместить всю логику из класса сервиса в Stateless EJB. В классе сервиса оставить только обращение к методам бизнес-интерфейса. EJB-компонент должен быть доступен удалённо (иметь Remote-интерфейс).
- Сформировать на уровне сервера приложений пул компонентов EJB настраиваемой мощности, динамически расширяемый при увеличении нагрузки.
- Установить ПО Consul и настроить Service Discovery с его помощью. Сервис должен регистрироваться в Service Discovery в момент запуска.

Изменения в "вызывающем" сервисе:

- Сконфигурировать окружение для работы сервиса на платформе Spring Boot.
- Запустить второй экземпляр сервиса на другом порту. Реализовать балансировку нагрузки между экземплярами с помощью Haproxy.

Оба веб-сервиса и клиентское приложение должны сохранить полную совместимость с API, реализованными в рамках предыдущих лабораторных работ.

Исходный код

https://github.com/xxPFFxx/SOA-lab3

Конфигурация сервиса в Consul с поддержанием в активном состоянии

```
@Singleton
@Startup
public class ConsulUtil {
    private AgentClient agentClientl;
    private String service_id;
    private Integer port;
```

```
    private String name;
    private Long ttl;

    @PostConstruct
    public void register() {
        try {
        ClassLoader classLoader = Thread.currentThread().getContextClassLoader();
        InputStream input = classLoader.getResourceAsStream("consul.properties");
        Properties properties = new Properties();
        properties.load(input);
        port = Integer.parseInt(properties.getProperty("consul.port"));
        name = properties.getProperty("consul.name");
        service_id = properties.getProperty("consul.service_id");
        ttl = Long.parseLong(properties.getProperty("consul.ttl"));

            Consul consul = Consul.builder().build();
            agentClient = consul.agentClient();
            agentClient.register(ImmutableRegistration.builder()
                    .id(service_id)
                    .name(name)
                    .port(port)
                    .check(Registration.RegCheck.ttl(ttl))
                    .build());
            System.out.println("Consul registered");
        }catch (Exception e){
            System.out.println("Error trying to connect to Consul");
        }
    }
    @Schedule(hour = "*", minute = "*", second = "*/20")
    public void checkIn() throws NotRegisteredException {
        agentClient.pass(service_id);
    }
}
```

Обращение к вызываемому сервису через Consul

```
@Configuration
public class ConfigurationUtil {
    @Value("${consul.main-app-name}")
    private String mainAppName;

    @Autowired
    private DiscoveryClient discoveryClient;

    private ServiceInstance serviceInstance = null;

    public ServiceInstance getServiceInstance(){
```

```
        return discoveryClient.getInstances(mainAppName).stream().findFirst().get();
    }

    public String urlApiService(){
        if (serviceInstance == null){
            serviceInstance = getServiceInstance();
        }
        return "https://"+ serviceInstance.getHost()+":"+serviceInstance.getPort();
    }
}
```

Настройка пула EJB

```
<glassfish-ejb-jar>
<enterprise-beans>
    <ejb>
        <ejb-name>HumanBeingService</ejb-name>
        <bean-pool>
            <max-pool-size>20</max-pool-size>
            <max-wait-time-in-millis>0</max-wait-time-in-millis>
            <steady-pool-size>1</steady-pool-size>
        </bean-pool>
    </ejb>
</enterprise-beans>
</glassfish-ejb-jar>
```

Вызов Remote EJB

```
public class RemoteBeanUtil {
    public static HumanBeingServiceInterface lookupRemoteStatelessBean() {
        Properties contextProperties = new Properties();
        contextProperties.setProperty(
                Context.INITIAL_CONTEXT_FACTORY,
                "com.sun.enterprise.naming.SerialInitContextFactory"
        );
        try {
            InitialContext context = new InitialContext(contextProperties);
            String appName = "global";
            String moduleName = "Ejb-1.0-SNAPSHOT";
            String beanName = "HumanBeingService";
            String lookupName = "java:" + appName + "/" + moduleName + "/" + beanName;
            return (HumanBeingServiceInterface) context.lookup(lookupName);
        } catch (NamingException e) {
            return new HumanBeingServiceInterface() {
                @SneakyThrows
                @Override
```

```
            public Response additionalTasks(String weaponTypeCount, String
weaponTypeArray, String uniqueImpactSpeed) {
                throw new EjbNotAvailableException("Не удалось получить доступ к EJB
HumanBeingService");
            }
            @SneakyThrows
            @Override
            public PagedHumanBeingList getHumanBeings(String perPage, String curPage,
String sortBy, String filterBy) {
                throw new EjbNotAvailableException("Не удалось получить доступ к EJB
HumanBeingService");
            }
            @SneakyThrows
            @Override
            public HumanBeing getHumanBeing(Long long_id) {
                throw new EjbNotAvailableException("Не удалось получить доступ к EJB
HumanBeingService");
            }
            @SneakyThrows
            @Override
            public void saveHumanBeing(String humanBeing) {
                throw new EjbNotAvailableException("Не удалось получить доступ к EJB
HumanBeingService");
            }
            @SneakyThrows
            @Override
            public void updateHumanBeing(String humanBeing, Long long_id) {
                throw new EjbNotAvailableException("Не удалось получить доступ к EJB
HumanBeingService");
            }
            @SneakyThrows
            @Override
            public void deleteHumanBeing(Long long_id) {
                throw new EjbNotAvailableException("Не удалось получить доступ к EJB
HumanBeingService");
            }
        };
    }
  }
}
```

Обработка ошибок Remote EJB

```
@Provider
public class EjbExceptionMapper implements ExceptionMapper<EJBException> {

    public Response toResponse(EJBException e) {
```

```
        Throwable nestedException = e.getCause().getCause().getCause();
        ExceptionDTO exceptionDTO = new ExceptionDTO();
        exceptionDTO.setMessage(nestedException.getMessage());
        if (nestedException instanceof BadRequestException){
            return Response
                    .status(Response.Status.BAD_REQUEST)
                    .entity(exceptionDTO)
                    .type(MediaType.APPLICATION_JSON)
                    .build();
        }else {
            return Response
                    .status(Response.Status.NOT_FOUND)
                    .entity(exceptionDTO)
                    .type(MediaType.APPLICATION_JSON)
                    .build();
        }
    }
}
```

Настройка Haproxy

Haproxy.cfg

```
frontend my_http_front
        bind *:8905
        option tcplog
        mode tcp
        default_backend my_http_back

backend my_http_back
        balance roundrobin
        mode tcp
        option ssl-hello-chk
        server myback1 localhost:8585 check weight 50
        server myback2 localhost:8586 check weight 50
```

Вывод

В ходе выполнения этой лабораторной работы я настроил регистрацию сервиса в Consul и последующее обнаружение этого сервиса. Переработал вызываемый сервис, переместив всю логику сервиса в Stateless EJB. Реализовал распределение нагрузки между вызывающими сервисами с помощью Haproxy.