Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Сервис-ориентированная архитектура

Лабораторная работа №3

Вариант 3002.1.3

Группа: P33122

Студент:

Беляков Д.С

Преподаватель:

Усков И. В.

Санкт-Петербург

2021

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:

- 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.
- Настроить второй экземпляр сервера приложений на другом порту, "поднять" на нём вторую копию веб-сервиса и пула EJB.
- Настроить балансировку нагрузки на оба запущенных узла через Haproxy.

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

Код: https://github.com/kevinche75/soa_lab3_itmo_autumn_2021

Добавление сервиса в Consul и поддержание его в активном состоянии

```
@Singleton
/*
```

```
 Registers the app in consul, also performs check in to consul every 15
seconds
*/
public class ServiceDiscoveryWorker {
   private Consul client = null;
   private static String serviceId = "1";

   {
       try {
           Context env = (Context)new
InitialContext().lookup("java:comp/env");
           serviceId = (String) env.lookup("serviceId");
           client = Consul.builder().build();
           AgentClient agentClient = client.agentClient();
           String serviceName = (String) env.lookup("serviceName");
           int port = (Integer) env.lookup("servicePort");
           System.out.println(serviceName);
           System.out.println(port);
           Registration service = ImmutableRegistration.builder()
                   .id(serviceId)
                   .name(serviceName)
                   .port(port)
                   .check(Registration.RegCheck.ttl(60L)) // registers with a
TTL of 3 seconds
                   .meta(Collections.singletonMap(
                           (String) env.lookup("serviceMetaUriKey"),
                           (String) env.lookup("serviceMetaUri"))
                   )
                   .build();

           agentClient.register(service);
           System.out.println("Service registered");
       } catch (Exception e) {
           System.err.println("Consul is unavailable");
       }
   }

   @SneakyThrows
   @Schedule(hour = "*", minute = "*", second = "*/15")
   public void checkIn() {
       AgentClient agentClient = client.agentClient();
       agentClient.pass(serviceId);
   }

}
```

Получение адреса через Consul

```
public static String getUriFromConsul() throws NamingException {
   Context env = (Context)new InitialContext().lookup("java:comp/env");
   String serviceName = (String) env.lookup("serviceName");
   String serviceMetaKey = (String) env.lookup("serviceMetaUriKey");
   if (client != null) {
```

```
        HealthClient healthClient = client.healthClient();
        List<ServiceHealth> nodes =
healthClient.getHealthyServiceInstances(serviceName).getResponse();
        if (nodes.size() > 0) {
            ServiceHealth service = nodes.get(0);
            System.out.println("Got service's 2 (Payara) uri from consul");
            String address = service.getNode().getAddress();
            int port = service.getService().getPort();
            String app = service.getService().getMeta().get(serviceMetaKey);
            return String.format("https://%s:%d/%s", address, port, app);
        }
    }
    System.err.println("Service 2 (Payara) not available from consul - using
default resource");
    return (String) env.lookup("uri");
}
```

Настройка пула EJB

```
<enterprise-beans>
    <ejb>
        <ejb-name>LabWorksService</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>
            <!--
<pool-idle-timeout-in-seconds>5</pool-idle-timeout-in-seconds>-->
            <!--
<disable-nonportable-jndi-names>true</disable-nonportable-jndi-names>-->
        </bean-pool>
    </ejb>
</enterprise-beans>
```

Вызов Remote EJB

```
Properties jndiProperties = new Properties();
jndiProperties.put(Context.INITIAL_CONTEXT_FACTORY,
"com.sun.enterprise.naming.SerialInitContextFactory");
try {
    Context env = (Context)new InitialContext().lookup("java:comp/env");
    final javax.naming.Context context = new InitialContext(jndiProperties);
    final String appName = (String) env.lookup("appName");
    final String moduleName = (String) env.lookup("moduleName");
    final String beanName = (String) env.lookup("beanName");
    final String viewClassName = LabWorkI.class.getName();
    final String scope = (String) env.lookup("scope");
```

```
   String lookupName = scope + ":" + appName + "/" + moduleName + "/" +
beanName + "!" + viewClassName;

   return (LabWorkI) context.lookup(lookupName);
} catch (NamingException e) {
   System.out.println("не получилось (");
   e.printStackTrace();
   return new LabWorkI() {

       @Override
       public ResponseWrapper getAllLabWorks(HashMap<String, String> map) {
           return new ResponseWrapper(500, "Server error, try again!");
       }

       @Override
       public ResponseWrapper getLabWork(String str_id) {
           return new ResponseWrapper(500, "Server error, try again!");
       }

       @Override
       public ResponseWrapper getMinName() {
           return new ResponseWrapper(500, "Server error, try again!");
       }

       @Override
       public ResponseWrapper countPersonalQualitiesMaximum(String str_pqm) {
           return new ResponseWrapper(500, "Server error, try again!");
       }

       @Override
       public ResponseWrapper createLabWork(String xmlStr) {
           return new ResponseWrapper(500, "Server error, try again!");
       }

       @Override
       public ResponseWrapper updateLabWork(String str_id, String xmlStr) {
           return new ResponseWrapper(500, "Server error, try again!");
       }

       @Override
       public ResponseWrapper deleteLabWork(String str_id) {
           return new ResponseWrapper(500, "Server error, try again!");
       }

       @Override
       public ResponseWrapper test() {
           return new ResponseWrapper(500, "Server error, try again!");
       }

       @Override
       public ResponseWrapper getLessMaximumPoint(HashMap<String, String>
map, String maximum_point) {
           return new ResponseWrapper(500, "Server error, try again!");
```

```
        }
    };
}
```

Настройка Haproxy

```
global
    log 127.0.0.1   local0
    log 127.0.0.1   local1 debug
    #log loghost    local0 info
    maxconn 4096

defaults
    log     global
    mode    http
    option  httplog
    option  dontlognull
    retries 3
    option redispatch
    maxconn 2000
    timeout connect      5000
    timeout client      50000
    timeout server      50000

 frontend stats
    bind *:8404
    stats enable
    stats uri /stats
    stats refresh 10s
    stats admin if LOCALHOST

frontend myfrontend
    bind *:7070 ssl crt localhost.pem
    default_backend myservers

backend myservers
    mode http
    balance roundrobin
    option forwardfor
    http-request set-header X-Forwarded-Port %[dst_port]
    http-request add-header X-Forwarded-Proto https if { ssl_fc }
    option httpchk HEAD / HTTP/1.1\r\nHost:localhost
    server server1 127.0.0.1:8181 ssl verify none
    server server2 127.0.0.1:8182 ssl verify none
```

Создание сертификата в формате PEM

```
keytool -export -alias payra -file payara.der -keystore payarastore
openssl x509 -inform der -in payara.der -out payara.pem
```

```
keytool -importkeystore -srckeystore payrastore -destkeystore
payarakeystore.p12 -deststoretype PKCS12
openssl pkcs12 -in payarakeystore.p12 -nodes -nocerts -out payara.key
cat payara.key >> localhost.pem
cat payara.pem >> localhost.pem
```

Вывод

В данной лабораторной работе я настроил регистрацию сервиса в Consul, переделал сервис с payara micro на payra, настроил EJB бины, а также перенаправление вызывающих сервисов через Haproxy, долго удивлялся магии сервисов, когда через день сервисы переставали работать.