Федеральное государственное автономное образовательное учреждение высшего образования

«**Университет ИТМО»**

**Факультет ПИиКТ**

**Дисциплина: Технологии веб-сервисов**

**Лабораторная работа 7**

Выполнили: Гурин Евгений Иванович, Камышанская Ксения Васильевна

Преподаватель: Дергачев Андрей Михайлович

Группа: Р4216

Санкт-Петербург 2024г.

# Задача

Требуется разработать приложение, осуществляющее регистрацию сервиса в реестре jUDDI, а также поиск сервиса в реестре и обращение к нему.

Рекомендуется реализовать консольное приложение, которое обрабатывает две команды. Итог работы первой команды – регистрация сервиса в реестре; вторая команда должна осуществлять поиск сервиса, а также обращение к нему.

# Выполнение

Данная лабораторная работа оказалась самой трудновыполнимой среди прочих. Трудности возникли в ряде аспектов. jUDDI является deprecated проектов, из-за чего так большие проблемы с документацией, а также используются старые javax спецификации, из-за которых возникает множество конфликтов в проекте с использованием актуальных Jakarta спецификаций. Особенно неприятными оказались проблемы на момент сборки итогового jar архива, в котором необходимо было иметь разные реализации jEE АПИ, которые не будут конфликтовать из-за разных версий АПИ. Помимо решения проблемы с конфликтующими реализациями существовала проблема с развертыванием jUDDI, для чего использовался Docker и созданием пользователей в jUDDI.

1. Для начала был создал модуль lab7-service-discovery для реализации методов регистрации и поиска сервисов в jUDDI

Для этого были созданы интерфейсы JuddiClient и ServiceDiscovery

|  |
| --- |
| /\*\*  \* jUDDI Client.  \*/  public interface JuddiClient {  /\*\*  \* Generate auth token.  \*  \* @return generated token  \* @throws RemoteException if got an error from jUDDI  \*/  AuthToken provideAuthToken() throws RemoteException;  /\*\*  \* Discard auth token after done.  \*  \* @param token auth token  \* @throws RemoteException if got an error from jUDDI  \*/  void discardAuthToken(AuthToken token) throws RemoteException;  /\*\*  \* Register business by name.  \*  \* @param businessName Name of business.  \* @param authToken auth token  \* @return business key of saved business  \* @throws RemoteException if got an error from jUDDI  \*/  String registerBusiness(String businessName, AuthToken authToken) throws RemoteException;  /\*\*  \* Register service by name and business key.  \*  \* @param businessKey jUDDI business key.  \* @param serviceName Name of service  \* @param authToken auth token  \* @return service key of saved service  \* @throws RemoteException if got an error from jUDDI  \*/  String registerService(String businessKey, String serviceName, AuthToken authToken) throws RemoteException;  /\*\*  \* Bind service url to exist service.  \*  \* @param serviceKey jUDDI service key  \* @param serviceUrl URL of service  \* @param authToken auth token  \* @return binding details  \* @throws RemoteException if got an error from jUDDI  \*/  BindingDetail bindService(String serviceKey, URL serviceUrl, AuthToken authToken) throws RemoteException;  /\*\*  \* Register both business and service by name.  \* Then bind service to URL.  \* Don't create business or service if already exists.  \*  \* @param businessName Name of business entity.  \* @param serviceName Name of service.  \* @param serviceUrl URL of service.  \* @return service key of saved service  \* @throws RemoteException if got an error from jUDDI  \*/  String registerBusinessService(String businessName, String serviceName, URL serviceUrl) throws RemoteException;  /\*\*  \* Find business key in jUDDI by name.  \*  \* @param businessName Name of business entity.  \* @return business key of found business  \* @throws RemoteException if got an error from jUDDI  \* @throws BusinessNotFound if business did not find  \*/  String findBusinessInfo(String businessName) throws RemoteException, BusinessNotFound;  /\*\*  \* Find service key in jUDDI by name and business key.  \*  \* @param businessKey business key  \* @param serviceName Name of service  \* @return service key of found service  \* @throws RemoteException if got an error from jUDDI  \* @throws ServiceNotFound if service did not find  \*/  String findServiceInfo(String businessKey, String serviceName) throws RemoteException, ServiceNotFound;  /\*\*  \* Find service binding by service key.  \*  \* @param serviceKey Service key  \* @return binding detail  \* @throws RemoteException if got an error from jUDDI  \*/  BindingDetail findServiceBindingInfo(String serviceKey) throws RemoteException;  /\*\*  \* Find service binding by business name and service name.  \*  \* @param businessName Name of business entity.  \* @param serviceName Name of service  \* @return binding detail  \* @throws RemoteException if got an error from jUDDI  \* @throws ServiceNotFound if service did not find  \* @throws BusinessNotFound if business did not find  \*/  BindingDetail findService(String businessName, String serviceName)  throws RemoteException, ServiceNotFound, BusinessNotFound;  } |

|  |
| --- |
| /\*\*  \* Service Discovery.  \*/  public interface ServiceDiscovery {  /\*\*  \* Register service and bind to URL by class.  \*  \* @param baseUrl base API url (without service path).  \* @param clazz service class.  \* @throws RemoteException if got an error on register  \*/  void registerService(String baseUrl, Class<?> clazz) throws RemoteException, MalformedURLException;  /\*\*  \* Find service bindings in jUDDI by service class.  \*  \* @param clazz service class.  \* @return list of registered service urls  \* @throws RemoteException if got an error from jUDDI  \* @throws BusinessNotFound if business did not find  \* @throws ServiceNotFound if service did not find  \*/  List<URL> findService(Class<?> clazz) throws RemoteException, BusinessNotFound, ServiceNotFound;  } |

1. Далее данные интерфейсы были реализованы

<https://github.com/GulDilin/itmo-web-services-technologies/blob/master/lab7/lab7-service-discovery/src/main/java/guldilin/discovery/JuddiClientImpl.java>

<https://github.com/GulDilin/itmo-web-services-technologies/blob/master/lab7/lab7-service-discovery/src/main/java/guldilin/discovery/ServiceDiscoveryImpl.java>

Было решено в качестве названия сервиса использовать название интерфейса, а для обозначения business name использовать кодовую строку tws

1. После этого на момент старта приложения была реализована автоматическая регистрация сервиса и прикрепление к URL

|  |
| --- |
| @PostConstruct  private void onInit() throws RemoteException, MalformedURLException {  this.serviceDiscovery.registerService(PropertyKey.APP\_URL.lookupValue(), CityService.class);  } |

1. Также был обновлен консольный клиент, в котором происходит поиск сервиса по business name и названию сервиса, а также происходит выбор рабочего access point на основе результата get запроса к wsdl схеме

|  |
| --- |
| /\*\*  \* Implementation of ServiceProvider.  \*/  public class ServiceProviderImpl implements ServiceProvider {  /\*\*  \* URL of city service url.  \*/  private final ServiceDiscoveryImpl serviceDiscovery;  /\*\*  \* Default Constructor.  \*  \* @param juddiHost jUDDI host name.  \* @param juddiPort jUDDI port.  \* @throws ConfigurationException for incorrect config.  \* @throws TransportException if jUDDI client cannot be created.  \*/  public ServiceProviderImpl(final String juddiHost, final Integer juddiPort)  throws ConfigurationException, TransportException {  this.serviceDiscovery = new ServiceDiscoveryImpl(new JuddiClientImpl(juddiHost, juddiPort));  }  private Boolean isUrlAlive(final URL url) {  try {  var client = HttpClient.newBuilder().build();  var request = HttpRequest.newBuilder(url.toURI()).GET().build();  var response = client.send(request, HttpResponse.BodyHandlers.ofString());  return response.statusCode() < Response.Status.BAD\_REQUEST.getStatusCode();  } catch (URISyntaxException | IOException | InterruptedException e) {  return false;  }  }  /\*\*  \* {@inheritDoc}  \*/  @Override  public CityWs provideCityService() throws RemoteException, BusinessNotFound, ServiceNotFound {  var urls = this.serviceDiscovery.findService(CityService.class);  var urlOptional = urls.stream().filter(this::isUrlAlive).findFirst();  if (urlOptional.isEmpty()) {  System.out.println("Not Found city service alive url");  throw new ServiceNotFound();  }  var url = urlOptional.get();  System.out.printf("Found city service alive url: %s\n", url);  var service = new CityService(url);  return service.getCityPort();  }  } |

Обновленный интерфейс консольного приложения

Usage: lab7-client.jar [options]

Options:

\* -c, -command

Command name

Possible Values: [find, create, update, patch, delete]

-help, -h

Show help message (also can be used with -c argument to show help

message for command)

\* -juddi\_host

jUDDI URL hostname (example: localhost)

\* -juddi\_port

jUDDI port (example: 8080)

# Вывод

В результате лабораторной работы был развернут jUDDI с использованием Docker. Были подключены библиотеки для доступа к jUDDI, реализованы классы для регистрации и поиска сервисов. Была реализована автоматическая регистрация сервиса в момент старта и поиск сервиса из консольного приложения. Было решено множество трудностей из-за того, что jUDDI уже выведен из эксплуатации и имеет не самую подробную документацию.