

Федеральное государственное автономное образовательное
учреждение высшего образования

Университет ИТМО

Дисциплина: Технологии веб-сервисов

Лабораторная работа 7

Выполнили:

Кривоносов Егор Дмитриевич

Группа: Р4214

Преподаватель:

Сафронов Андрей Геннадьевич

2025 г.

Санкт-Петербург

Оглавление

Техническое задание	3
Постановка задачи	3
Этапы выполнения	3
Комментарии к архитектурным и функциональным аспектам реализации	4
Ссылка на код	7
Вывод	7

Техническое задание

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.

Постановка задачи

1. Настроить локальный сервер jUDDI для регистрации и поиска сервисов.
2. Разработать клиентское приложение для взаимодействия с jUDDI через стандартные API.
3. Реализовать команды регистрации сервиса и поиска зарегистрированных сервисов.
4. Обеспечить корректную работу приложения на Java 8, включая настройку переменных среды и конфигурации сервера.

Этапы выполнения

1. **Скачивание и настройка jUDDI:**
 - Скачан архив с jUDDI версии 3.3.9 с официального сайта Apache.
 - Убедился, что путь к jUDDI не содержит русских символов и пробелов.
 - Установлены переменные окружения:
 - `JAVA_HOME` и `JRE_HOME` на версию Java 8.
 - Внесены изменения в `catalina.bat`:
 - `set JAVA_OPTS=%JAVA_OPTS% -Djavax.xml.accessExternalDTD=all`
 - Запуск сервера выполнен через `startup.bat` в папке `juddi-tomcat-{version}/bin`.
 - Авторизация: `username="uddiadmin", password="da_password1"`.
2. **Разработка клиента:**
 - Реализован класс `JuddiClient`, обеспечивающий:
 - Аутентификацию в системе jUDDI.
 - Регистрацию бизнеса и сервисов.
 - Поиск бизнеса и сервисов.
 - Получение URL-адресов зарегистрированных сервисов.

- Для упрощения работы с API созданы вспомогательные методы, такие как `registerService`, `findServiceInfo`, `bindService` и другие.
3. **Реализация консольного интерфейса:**
- Созданы команды для регистрации сервиса и поиска сервисов с последующим выводом их URL.
4. **Тестирование:**
- Проведена регистрация тестового сервиса в реестре.
 - Выполнен поиск зарегистрированного сервиса и успешное обращение к нему.

Комментарии к архитектурным и функциональным аспектам реализации

Регистрация бизнеса и сервиса:

Эти методы позволяют зарегистрировать бизнес и связанные с ним сервисы в реестре, используя ключи авторизации.

```
public String registerBusiness(String businessName, AuthToken authToken) throws RemoteException {
    BusinessEntity businessEntity = new BusinessEntity();
    businessEntity.getName().add(new Name(businessName, Locale.ENGLISH.getDisplayLanguage()));
    SaveBusiness sb = new SaveBusiness();
    sb.getBusinessEntity().add(businessEntity);
    sb.setAuthInfo(authToken.getAuthInfo());
    return this.publisher.saveBusiness(sb).getBusinessEntity().get(0).getBusinessKey();
}

public String registerService(String businessKey, String serviceName, AuthToken authToken) throws RemoteException {
    BusinessService service = new BusinessService();
    service.setBusinessKey(businessKey);
    service.getName().add(new Name(serviceName, Locale.ENGLISH.getDisplayLanguage()));
    SaveService ss = new SaveService();
    ss.getBusinessService().add(service);
    ss.setAuthInfo(authToken.getAuthInfo());
    return this.publisher.saveService(ss).getBusinessService().get(0).getServiceKey();
}
```

Поиск сервиса:

Этот метод возвращает ключ зарегистрированного сервиса, если он найден в реестре.

```
public String findServiceInfo(String businessKey, String serviceName) throws RemoteException {
    FindService findServiceData = new FindService();
    findServiceData.setBusinessKey(businessKey);
    findServiceData.getName().add(new Name(serviceName, Locale.ENGLISH.getDisplayLanguage()));
    ServiceList serviceList = this.inquire.findService(findServiceData);
    if (serviceList == null || serviceList.getServiceInfos() == null
        || CollectionUtils.isEmpty(serviceList.getServiceInfos().getServiceInfo())) {
        return null;
    }

    List<ServiceInfo> serviceInfos = serviceList.getServiceInfos().getServiceInfo();
    return serviceInfos.get(0).getServiceKey();
}
```

```
}
```

Получение URL-адресов из привязки сервиса:

Метод позволяет получить список URL-адресов, связанных с зарегистрированным сервисом.

```
public static List<String> getUrlsFromBinding(BindingDetail bindingDetail) {  
    List<String> urls = new ArrayList<>();  
    for (BindingTemplate bindingTemplate : bindingDetail.getBindingTemplate()) {  
        AccessPoint accessPoint = bindingTemplate.getAccessPoint();  
        urls.add(accessPoint.getValue());  
    }  
  
    return urls;  
}
```

Конфигурация клиента jUDDI

Файл **uddi.xml** содержит конфигурацию клиента jUDDI, включая информацию о транспортном уровне, URL-адресах и настройках узла.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>  
<uddi>  
    <reloadDelay>5000</reloadDelay>  
    <client name="example-manager">  
        <nodes>  
            <node isHomeJUDDI="true">  
                <name>default</name>  
                <clientName>default</clientName>  
                <description>jUDDI node</description>  
                <proxyTransport>  
                    org.apache.juddi.v3.client.transport.JAXWSTransport  
                </proxyTransport>  
  
                <custodyTransferUrl>http://${serverName}:${serverPort}/juddiv3/services/custody-transfer</custodyTransferUrl>  
  
                <inquiryUrl>http://${serverName}:${serverPort}/juddiv3/services/inquiry</inquiryUrl>  
  
                <inquiryRESTUrl>http://${serverName}:${serverPort}/juddiv3/services/inquiryRest</inquiryRESTUrl>  
  
                <publishUrl>http://${serverName}:${serverPort}/juddiv3/services/publish</publishUrl>  
  
                <securityUrl>http://${serverName}:${serverPort}/juddiv3/services/security</securityUrl>  
  
                <subscriptionUrl>http://${serverName}:${serverPort}/juddiv3/services/subscription</subscriptionUrl>  
  
                <subscriptionListenerUrl>http://${serverName}:${serverPort}/juddiv3/services/subscription-listener</subscriptionListenerUrl>  
            </node>  
        </nodes>  
    </client>  
</uddi>
```

reloadDelay: Указывает интервал обновления конфигурации клиента (в миллисекундах). В данном случае — каждые 5000 мс (5 секунд).

proxyTransport: Указывает транспортный слой. Используется **JAXWSTransport**, который работает на основе протокола SOAP.

URL-адреса: Динамически генерируются с использованием переменных `${serverName}` и `${serverPort}`, что позволяет гибко изменять настройки без необходимости редактирования самого файла.

- **custodyTransferUrl:** Для передачи владения.
- **inquiryUrl:** Для запросов к реестру.
- **publishUrl:** Для публикации данных в реестре.
- **securityUrl:** Для аутентификации и получения токенов.

isHomeJUDDI: Указывает, что данный узел является локальным (домашним).

The screenshot displays the JAX-WS Transport console interface. At the top, there is a navigation bar with links for Home, Discover, Create, Settings, and Help, along with a login section. The main content area is titled 'Businesses' and shows a table of services. The table has columns for Name, Service, and a 'Show' button. The first service listed is 'An Apache JUDDI Node' with a 'Show 13' button and an 'Add' button. The second service is 'TVS' with a 'Show 1' button and an 'Add' button. Below the table, there is a 'Service Editor' section. The 'Service Editor' has a 'Binding Template' tab selected. It shows the 'Business Key' and 'Service Key' fields, both containing long alphanumeric strings. Below these, there is a 'Binding Template Key' field, also containing a long alphanumeric string. The 'Access Point Type' is 'wsdlDeployment' and the 'Access Point' is 'http://localhost:8081'. At the bottom, there is a row of buttons: Save, Delete, Digitally Sign, Subscribe, Transfer Ownership, and View as XML.

Businesses

Name	Service
An Apache JUDDI Node	Show 13 Add
TVS	Show 1 Add

Service Editor

Business Key - The Business Key is the unique Identifier for this business and exists within this registry. [Help](#)
uddi:juddi.apache.org:4dd6d218-101a-4f40-a8fd-fb8d342a278e

Service Key - The Service Key is the unique Identifier for this service. If you specify a service key, it must be prefixed with an existing partition (key generator).
uddi:juddi.apache.org:a1e13d19-0a07-442b-bd7e-840b04883308

General **Categories** **Binding Template** **Signatures** **Operational Info**

Binding Template - A service in UDDI really defines a specific type of service, not necessarily an implementation of a service. Binding templates define specifically an implementation of a service and normally includes an access point describing how to use the service. Each service may have 0 or more binding templates. Some registries impose limits on the number of binding templates per service.
[Add a Binding Template](#)

Binding Template Key	Access Point Type	Access Point
uddi:juddi.apache.org:7626f6f0-f0e4-4143-8eb1-27949e330fb2	wsdlDeployment	http://localhost:8081

[Save](#) [Delete](#) [Digitally Sign](#) [Subscribe](#) [Transfer Ownership](#) [View as XML](#)

Ссылка на код

<https://github.com/RedGry/TVS-LABS/tree/lab7>

Вывод

В ходе работы была создана система для взаимодействия с реестром jUDDI. Разработанное приложение успешно выполняет задачи регистрации сервисов, поиска и получения их URL-адресов. Работа над проектом позволила изучить возможности jUDDI и его интеграцию с Java-приложениями. Итоговый результат соответствует требованиям технического задания.