Web 服务计算技术实验报告

实验选题：服务注册 UDDI

SX1916115 张靖棠

2020.05

目录

UDDI

统一描述发现和集成 (UDDI) 提供一种发布和查找服务描述的方法。UDDI 数据实体提供对定义业务和服务信息的支持。WSDL 中定义的服务描述信息是 UDDI 注册中心信息的补充。UDDI 提供对许多不同类型的服务描述的支持。因此，UDDI 没有对 WSDL 的直接支持， 也没有对任何其它服务描述机制的直接支持。

Web 服务描述语言 (WSDL) 是用于描述 Web 服务的一种 XML 语言，它将 Web 服务描述为一组对消息进行操作的网络端点。一个 WSDL 服务描述包含对一组操作和消息的一个抽象定义，绑定到这些操作和消息的一个具体协议，和这个绑定的一个网络端点规范。

UDDI 组织，即 UDDI.org，已经发布了一个优化方法文档，标题为 在 UDDI 注册中心 1.05 中使用 WSDL (请参阅 参考资料)。这个优化方法文档描述了关于如何在 UDDI 注册中心发布 WSDL 服务描述的一些元素。本文的目的就是增加这种信息。主要的焦点问题是如何将一个完整的 WSDL 服务描述映射到 UDDI 注册中心，现有的 WSDL 工具和运行时环境要求必须做到这一点。本文中的信息遵守那个优化方法文档中列出的大致过程， 且与 WSDL 1.1、UDDI 1.0 和 UDDI 2.0 规范一致。

jUDDI

jUDDI 是 UDDI 的 Java 实现的私有 UDDI 注册中心，是一个开源项目，包含

了对 UDDI v3 标准的服务端和客户端实现。能够独立于操作系统平台，运行于任何支持 Servlet 2.3 标准的 Java 应用服务器上，且支持集群部署。在实验中，我使用了 jUDDI 目前最新的发布版 (Mar 15, 2020, jUDDI Release 3.3.8) 进行注册中心的部署。并通过 jUDDI 提供的 API，通过编程的方式，实现服务的注册发布、检索、发现、删除、调用等功能。

jUDDI 数据模型

想要在 jUDDI 上注册服务，首先需要了解 UDDI 的数据模型。根据 UDDI 标准，数据被组织为一种层次化的结构。最顶层的结构是 Business Entities，它们包含了若干 Business Services，而 Business Services 包含了若干个 Binding Templates。因此，如果想要发布一个 Business Service，必须要先有一个 Business Entity；如果想要发布一个 Binding Template，就必须要现有一个 Business Service。

Business Entities 描述的是一个对其发布的所有服务负责的组织单元，主要包含一些描述信息和联系人信息。比如说，一个小公司可以是一个 Business Entity；一个大公司的每一个部门都可以是一个 Business Entity。

Business Service 是 SOA 架构的组成部分，代表了能够被客户端使用的功能单元。在 UDDI 中，这一部分主要包含一些描述服务的信息 (如服务名称、服务描述、服务分类等)。而服务实际的技术细节位于 Binding Templates 中。

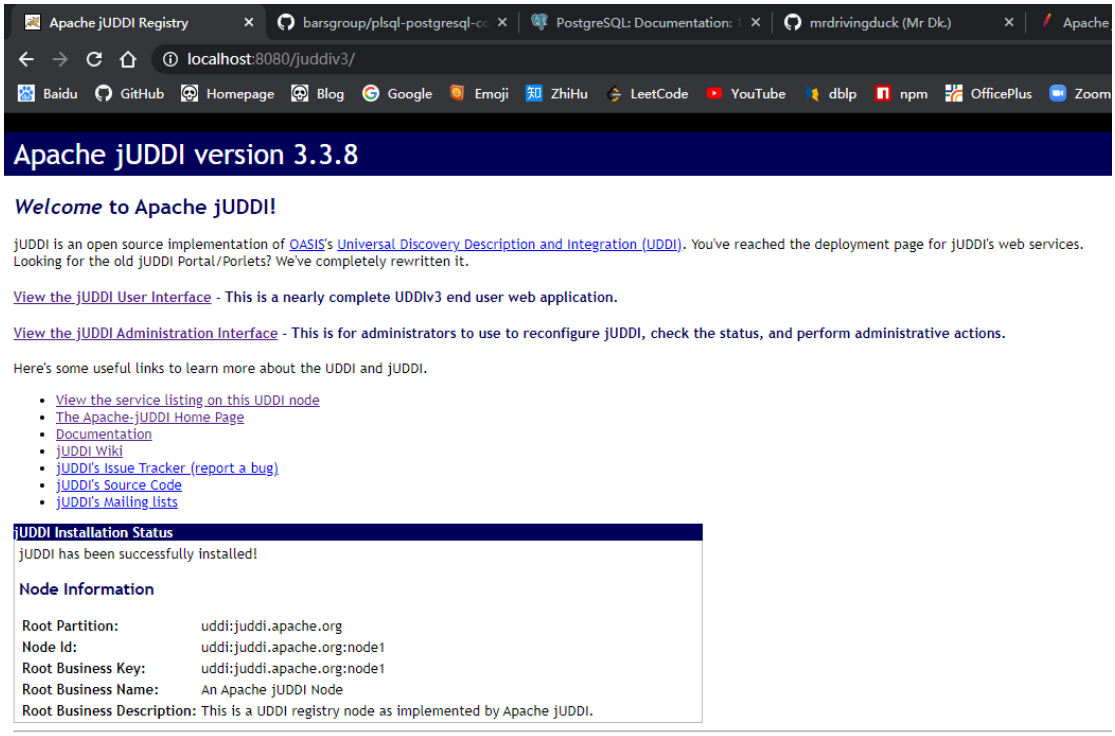
Binding Templates 中描述了服务的接入点、服务的 WSDL 位置等信息。客户端通过这些信息，就能得知去哪里使用服务、如何使用服务。

以上，UDDI 标准只定义了如何注册一个服务，但没有定义谁能够注册服务，

即忽略了认证的问题。jUDDI 中引入了一个称为 Publisher 的概念。只有在 jUDDI 注册中注册过的 Publisher，才能够向 jUDDI 注册服务。在访问 jUDDI 之前，使用者需要向 jUDDI 出示 Publisher 用户名和密码进行身份认证。随后，对合法的 Publisher，jUDDI 会返回一个授权 token。有这个 token 之后，才能对上述的其它数据模型进行访问。

jUDDI 部署

jUDDI 可以部署于任意 Servlet 容器上。根据我选择的 jUDDI 版本 (3.3.8, March 15, 2020)，我选择将其部署在 Tomcat 9 上。开始运行 Tomcat 后，访问 localhost:8080，就能够进入 jUDDI 的 Web 页面。



jUDDI 的 Web 页面分为两部分。一部分是管理员页面，用于对整个服务注册中心进行管理，需要在 Tomcat 中配置好的管理账户才能登录进入该页面。另

一部分是用户 UI 界面，确切地说就是一个客户端 GUI。用户可以通过这个 GUI 来进行服务发现、服务注册等。

在本实验的后续过程中，主要就是利用 jUDDI 提供的 API，以编程的方式，来完成 GUI 所能完成的功能。即用户可以通过在 GUI 中填写表单来注册、检索、发现服务，也可以用编程的方式完成相同的功能。

服务注册功能

服务注册主要分为两步。第一步是在某个 endpoint 上运行要被注册的服务，第二步是向 jUDDI 注册这个 endpoint。

作为第一步，我借用了一个 jUDDI 官方给出的示例 Web service。其功能非常简单，用户以一个字符串作为参数向这个 Web service 发送 request，Web service 将这个字符串的前面附上 "hello" 后作为 response 返回。这个 Web service 被部署到了 localhost 的一个端口上。核心代码如图所示：

```
You, a few seconds ago | 1 author (You)
@WebService(endpointInterface = "org.apache.juddi.samples.HelloWorld", serviceName = "HelloWorld")

public class HelloWorldImpl implements HelloWorld {

    public String sayHi(String text) {
        // System.out.println("sayHi called");
        return "Hello " + text;
    }

}
```

第二步是向 UDDI 注册这个服务。首先，在上述章节中已经提到，只有合法的 Publisher 才能向 jUDDI 注册服务。在 jUDDI 中，已经内置了一个名为 root 的管理员 Publisher。因此要先使用 root 注册一个新的合法的 Publisher，然后再用这个新注册的 Publisher 来注册服务。

根据这个步骤，首先使用 root 身份进行认证，拿到 token。使用这个 token，调用 jUDDI 的 savePublisher API，创建了一个新的 Publisher。关于官方文档的说明，一个 Publisher 必须要有一个 KeyGenerator 才能使用。因此还需要通过 API 来创建 KeyGenerator。

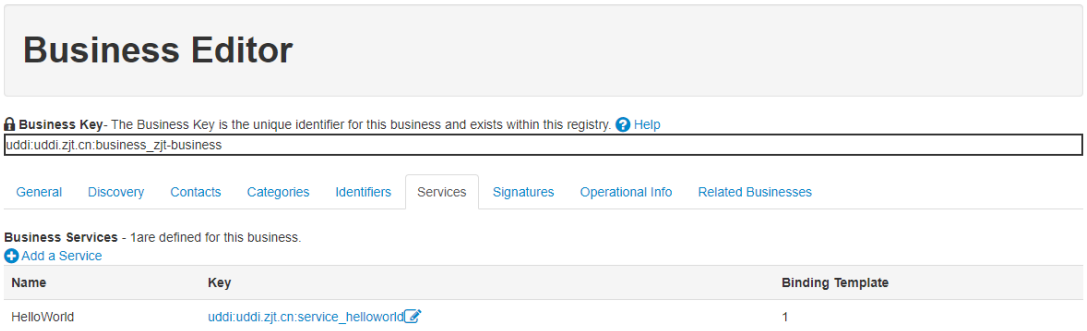
关于服务的一些描述信息，如名称、域等信息，编写在一个特定格式的 XML 文件里：

```
You, 2 days ago | 1 author (You)
<?xml version="1.0" encoding="ISO-8859-1" ?>
<uddi xmlns="urn:juddi-apache-org:v3_client" xsi:schemaLocation="classpath:/xsd/uddi-client.xsd">
  <reloadDelay>5000</reloadDelay>
  <client name="example-client" callbackUrl="http://localhost:8079/subscriptionlistener_uddi_client">
    <nodes>
      <node>
        <!-- required 'default' node -->
        <name>default</name>
        <properties>
          <property name="serverName" value="localhost" />
          <property name="serverPort" value="8080" />
          <property name="keyDomain" value="uddi.zjt.cn" />
          <property name="businessName" value="ZJT-business" />
          <!-- for UDDI nodes that use HTTP u/p, using the following
          <property name="basicAuthUsername" value="root" />
          <property name="basicAuthPassword" value="password" />
          <property name="basicAuthPasswordIsEncrypted" value="false" />
          <property name="basicAuthPasswordCryptoProvider" value="org.apache.juddi.v3.client.crypto.AE
```

使用这个 XML 文件作为配置，实例化一个 UDDI 客户端。另外，实例化一个新的 Business。将服务的具体描述信息填写到这个 Business Entity 对象中，然后调用客户端进行注册：

```
BusinessEntity myBusEntity = new BusinessEntity();
Name myBusName = new Name();
myBusName.setValue(businessName);
myBusEntity.getName().add(myBusName);
myBusEntity.setBusinessKey("uddi:" + keyDomain + ":business_" + businessName);
clerk.register(myBusEntity);
```

登录 jUDDI 的 Web GUI，可以看到服务被成功注册了：



服务检索功能

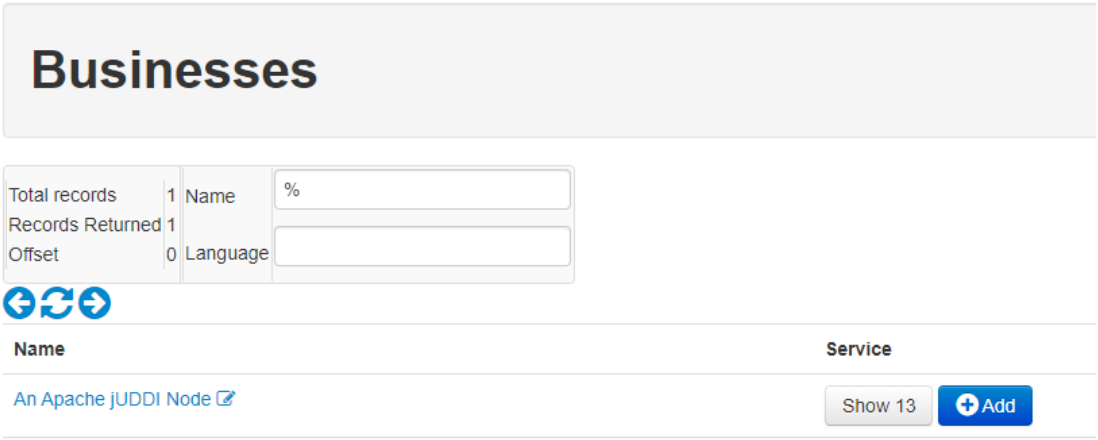
使用了与上述服务注册相同的服务配置文件，用于检索经过上述步骤成功注册的服务。

同样，使用配置文件实例化一个 UDDI 客户端，然后根据配置文件中的服务名，拼接出这个 Business 在 jUDDI 中的唯一的 key。根据这个 Business Key，就可以检索到已经注册的服务：

服务删除功能

服务删除与服务检索类似，都是指定一个 Business 或 Service 或 Binding Template 的唯一的 key，然后调 API 中的 unregister 函数。调用完成后，登录 Web GUI 查

看，通过上述步骤注册的 Business 不见了：



服务发现功能

服务发现与服务检索的区别在于，使用服务检索时，已经确定了要检索的服务（key）；而使用服务发现时，并不知道目前注册库中有哪些服务。函数会返回一个服务列表，然后通过对这个列表进行遍历输出，查看目前注册库中的所有服务。

在进行服务发现时，首先还是要用合法的 Publisher 进行认证得到 token，然后以 token 作为凭据，实例化一个 FindBusiness 对象，从而获得所有的 Business。

上述章节已经提到，UDDI 标准的数据模型是层次化的。因此，得到所有 Business 之后，还要依次获取每个 Business 下属的所有 Service；对每个 Service，依次获取其下属的所有 Binding Template。大致的代码流程如下：

```java
Run | Debug
public static void main(String[] args) {
    Discover discover = new Discover();
    try {

        String token = discover.getAuthKey("uddi", "uddi");
        BusinessList findBusiness = discover.getBusinessList(token);
        discover.printBusinessInfo(findBusiness.getBusinessInfos());
        discover.printBusinessDetails(findBusiness.getBusinessInfos(), token);
        discover.printServiceDetailsByBusiness(findBusiness.getBusinessInfos(), token);

        security.discardAuthToken(new DiscardAuthToken(token));

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

服务调用功能

这部分其实不属于服务注册的范畴，我只是用它来测试一下注册后的服务能否使用。同样，使用了服务注册时使用的配置文件，里面描述了 Publisher 信息，用于实例化一个 ServiceLocator。ServiceLocator 根据已注册服务唯一的 key，查询该服务的 endpoint，即 Web service 实际的地址。根据 Web service 在 WSDL 中定义的 request 格式，向这个地址发送请求，并得到了对应的 response。服务调用成功。

```
四月 30, 2020 11:02:22 下午 org.apache.juddi.v3.client.config.UDDIClerk register
信息: Registering subscription with key uddi:uddi.zjt.cn:service_cache_localhost
Add Listener to Cache in 6932 [milliseconds]
1. UDDI Lookup - Elapsed time: 532[milliseconds] Endpoint=http://localhost:18080/services/helloworld
2. Cache Lookup - Elapsed time: 0[milliseconds] Endpoint=http://localhost:18080/services/helloworld
********** Service reply: Hello Mr Dk.
```

调用成功的前提是，服务必须已经在其向 jUDDI 注册的 endpoint 上运行。

源代码

https://github.com/mrdrivingduck/uddi-experiment

参考资料

https://juddi.apache.org/docs/3.3/juddi-client-guide/html_single/#_uddi_data_model
https://juddi.apache.org/docs/3.3/juddi-client-guide/html_single/#_use_case_wsdl
https://juddi.apache.org/docs/3.3/juddi-client-guide/html_single/#_simple_publishing_example
https://juddi.apache.org/docs/3.3/juddi-guide/html/ch01.html
https://juddi.apache.org/docs/3.3/juddi-guide/html/ch02.html