国际物流云商系统第十二天

一．便民网站分析

1.需求

我们经常通过便民网站，查询日常信息，例如天气预报、手机归属地查询等，那这些功能是如何实现的呢？

类似天气、手机归属地这样的信息在我们的数据库中是不存在的，必须通过远程调用来获取这些信息，那开发这样的功能， 有哪些解决方案呢？

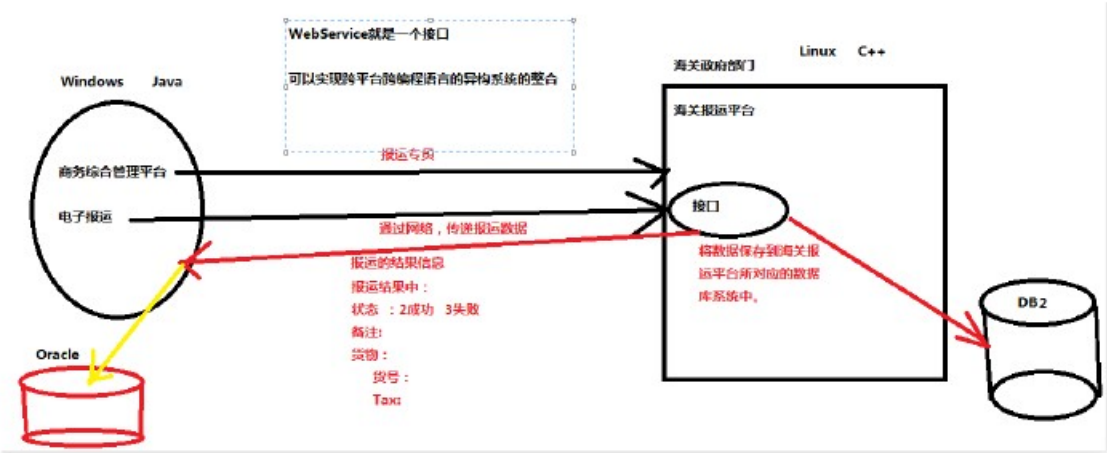
2.跨服务器调用技术

（1）Socket（服务端:提供服务的一端,开启服务 客户端：调用服务的一端，连接服务）

（2）通过 http 调用实现（客户端通过 HttpClient 调用）

（3）**Webservice**

Web service 即 web 服务，它是一种跨编程语言和跨操作系统平台的远程调用技术即跨平台远程调用技术。



（4）RMI (远程方法调用) java 调用 java　C# remoting

二．Webservice 规范及三要素

1.开发 webService 三种规范

JAVA 中共有三种 WebService 规范，分别是 JAX-WS（JAX-RPC）、JAXM&SAAJ、JAX-RS。

(1)Jaxws

JAX-WS 的全称为 Java API for XML-Based Webservices ，早期的基于 SOAP 的 JAVA 的 Web 服务规范 JAX-RPC（Java API For XML-Remote Procedure Call）目前已经被 JAX-WS 规范取代。从 java5 开始支持 JAX-WS2.0 版本，Jdk1.6.0_13 以后的版本支持 2.1 版本，jdk1.7 支持 2.2 版本。

Jaxws 开发的 webservice 传输 soap 协议。

特点：开发简单，高度封装实现细节。

(2)JAXM&SAAJ

JAXM（JAVA API For XML Message）主要定义了包含了发送和接收消息所需的 API，

SAAJ（SOAP With Attachment API For Java，JSR 67）是与 JAXM 搭配使用的 API，为构建 SOAP 包和解析 SOAP 包提供了重要的支持，支持附件传输等，JAXM&SAAJ 与 JAX-WS 都是基于 SOAP 的 Web 服务，相比之下 JAXM&SAAJ 暴漏了 SOAP 更多的底层细节，编码比较麻烦，而 JAX-WS 更加抽象，隐藏了更多的细节，更加面向对象，实现起来你基本上不需要关心 SOAP 的任何细节。

(3)JAX-RS

JAX-RS 是 JAVA 针对 REST(Representation State Transfer)风格制定的一套 Web 服务规范，由于推出的较晚，该规范（JSR 311，目前 JAX-RS 的版本为 1.0）并未随 JDK1.6 一起发行。

需要大家重视，当前比较流行，因为 jax-RS 可以发布 REST 风格 webservice，因为 REST 的 webservice 不采用 soap 传输，直接采用 http 传输，可以返回 xml 或 json，比较轻量。

2.WebService 三要素

（1）SOAP

soap 协议是 webservice 的传输协议，即**简单对象访问协议**。Soap 协议是 xml 格式，理解为基于 http 传输 xml 数据。Soap=http+xml

特点：1.跨平台　2：跨语言　3：w3c 指定的标准

各个开发语言都按照相同的标准实现 webservice 的开发。

（2）WSDL

WSDL 是 webservice 的使用说明书。

WSDL 指网络服务描述语言(Web Services Description Language)，是一种使用 XML 编写的文档，用于描述网络服务，也可用于定位网络服务。

WSDL 是 webservice 的使用说明书。根据 wsdl 去如何调用 webservice。从下往上读，找到 service 服务视图，通过 binding 找到 portType（服务类）。

```
<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
xmlns:wsp="http://www.w3.org/ns/ws-policy" xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://lb.itcast.cn/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://lb.itcast.cn/" name="WeatherService">
▼<types>
  ▼<xsd:schema>
     <xsd:import namespace="http://lb.itcast.cn/" schemaLocation="http://localhost:12345/weather?xsd=1"/>
   </xsd:schema>
 </types>
▼<message name="getInfo">
   <part name="parameters" element="tns:getInfo"/>
 </message>
▼<message name="getInfoResponse">
   <part name="parameters" element="tns:getInfoResponse"/>
 </message>
▼<portType name="Weather">
  ▼<operation name="getInfo">
     <input wsam:Action="http://lb.itcast.cn/Weather/getInfoRequest" message="tns:getInfo"/>
     <output wsam:Action="http://lb.itcast.cn/Weather/getInfoResponse" message="tns:getInfoResponse"/>
   </operation>
 </portType>
▼<binding name="WeatherPortBinding" type="tns:Weather">
   <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
  ▼<operation name="getInfo">
     <soap:operation soapAction=""/>
    ▼<input>
       <soap:body use="literal"/>
     </input>
    ▼<output>
       <soap:body use="literal"/>
```

（3）UDDI(了解)

UDDI 统一描述、发现和集成协议是一个目录服务，存储了全球的 webservice 地址。

UDDI 旨在将全球的 webservcie 资源进行共享，促进全球经济合作。

三．CXF 框架

1.什么是 CXF

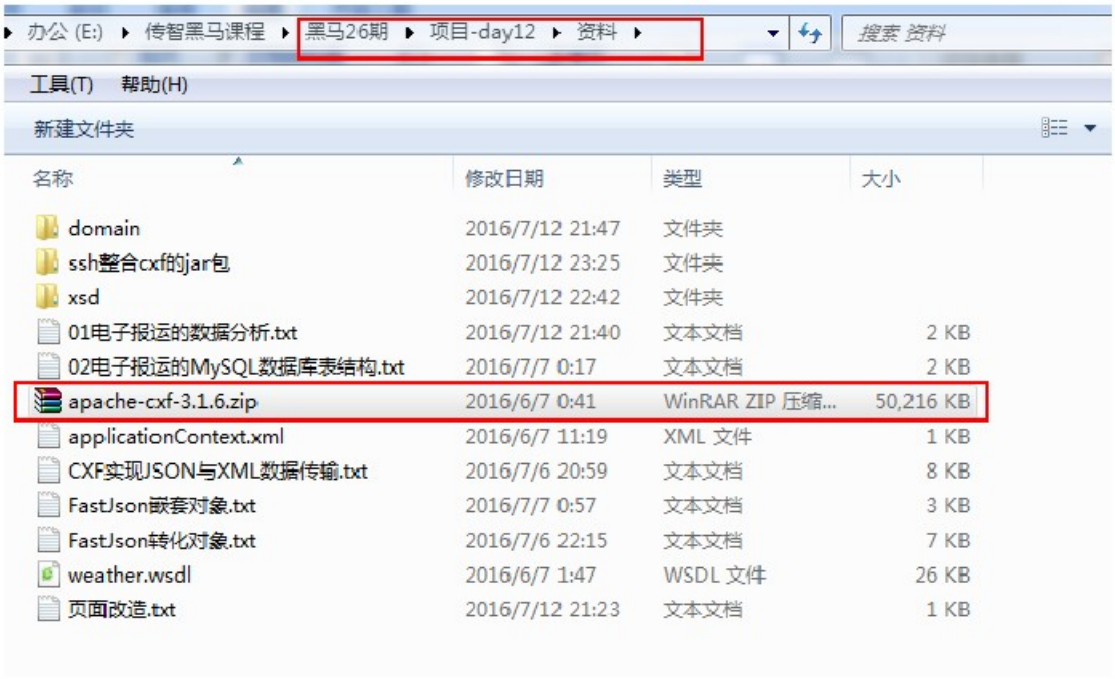
CXF，apache 下的 webservice 的开源框架。

Apache CXF = Celtix + Xfire，开始叫 Apache CeltiXfire，后来更名为 Apache CXF 了，以下简称为 CXF。Apache CXF 是一个开源的 web Services 框架，CXF 帮助您构建和开发 web Services ，它支持多种协议，比如：SOAP1.1,1,2、XML/HTTP、REST

HTTP 或者 CORBA。

灵活的部署：可以运行有 Tomcat,Jboss,weblogic,Jetty(内置)上面。

下载 cxf3.1.6 版本。



2.JAX-WS 方式开发服务端

开发步骤：

(1).建立工程，引入 CXF 的 JAXWS 方式的坐标

```
<!-- 要进行jaxws 服务开发 -->
<dependency>
    <groupId>org.apache.cxf</groupId>
    <artifactId>cxf-rt-frontend-jaxws</artifactId>
    <version>3.0.1</version>
</dependency>
<!-- 内置jetty web服务器 -->
<dependency>
    <groupId>org.apache.cxf</groupId>
    <artifactId>cxf-rt-transports-http-jetty</artifactId>
    <version>3.0.1</version>
</dependency>
<!-- 日志实现 -->
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-log4j12</artifactId>
    <version>1.7.12</version>
```

```
</dependency>
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.10</version>
    <scope>test</scope>
</dependency>
```

(2)添加 domain 类

```
public class Car {
    private Integer id;
    private String carName;
    private Double price;
```

```
public class User {
    private Integer id;
    private String username;
    private String city;
```

(3)创建服务类，添加注解@WebService

添加服务接口，需要在接口上加入注解@WebService，针对接口中的方法也可以使用@WebMethod 注解，但也可以不写。

```
@WebService
public interface IUserService {
    public String sayHi(String name);
    public List<Car> findCarsByUser(User user);
}
```

添加服务实现类 UserServiceImpl

```
public class UserServiceImpl implements IUserService {
    public String sayHi(String name) {
        System.out.println("hi,"+name);
        return "hi,"+name;
    }
    public List<Car> findCarsByUser(User user) {
        if ("宋江".equals(user.getUsername())) {
            List<Car> cars = new ArrayList<Car>();
            Car car1 = new Car();
            car1.setId(1);
            car1.setCarName("大众途观");
            car1.setPrice(200000d);
            cars.add(car1);

            Car car2 = new Car();
            car2.setId(2);
            car2.setCarName("现代ix35");
            car2.setPrice(170000d);
            cars.add(car2);

            return cars;
        } else {
            return null;
        }
    }
```

(4)编写服务发布的测试类，实现 webservice 的发布

```
public class WsServerTest {
    public static void main(String[] args) {
        //1.添加JaxWs服务发布工厂
        JaxWsServerFactoryBean factory = new JaxWsServerFactoryBean();
        //2.设置服务发布地址
        factory.setAddress("http://localhost:12345/user");
        //3.设置服务的实现类对象
        factory.setServiceBean(new UserServiceImpl());
        //4.发布服务
        factory.create();
    }
}
```

测试访问地址：http://localhost:12345/user?wsdl 就可以看到访问结果为 xml

```
<?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl
  xmlns:tns="http://service.itcast.cn/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:ns1="http://schemas.xmlsoap.org/soap/http" name="UserServiceImplService"
  targetNamespace="http://service.itcast.cn/">
  - <wsdl:types>
    - <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://service.itcast.cn/"
      elementFormDefault="unqualified" targetNamespace="http://service.itcast.cn/" version="1.0">
      <xs:element name="findCarsByUser" type="tns:findCarsByUser" />
      <xs:element name="findCarsByUserResponse" type="tns:findCarsByUserResponse" />
      <xs:element name="sayHi" type="tns:sayHi" />
      <xs:element name="sayHiResponse" type="tns:sayHiResponse" />
      - <xs:complexType name="findCarsByUser">
```

3.JAX-WS 方式开发客户端

开发步骤：

（1）建立工程，导入 Maven 坐标

```
<!-- 要进行jaxws 服务开发 -->
        <dependency>
            <groupId>org.apache.cxf</groupId>
            <artifactId>cxf-rt-frontend-jaxws</artifactId>
            <version>3.0.1</version>
        </dependency>

        <!-- 内置jetty web服务器 -->
        <dependency>
            <groupId>org.apache.cxf</groupId>
            <artifactId>cxf-rt-transports-http-jetty</artifactId>
            <version>3.0.1</version>
        </dependency>

        <!-- 日志实现 -->
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-log4j12</artifactId>
            <version>1.7.12</version>
        </dependency>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.10</version>
            <scope>test</scope>
        </dependency>
```

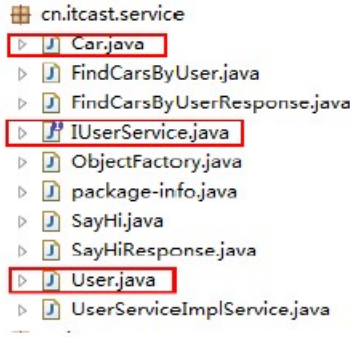
（2）wsimport 生成本地代码

它是在 JDK 安装目录的 bin 下

它的作用是将 wsdl 说明书，翻译成 java 代码

```
Wsimport –s . http://localhost:12345/user?wsdl
```

可以看到它会在指定的当前目录下，生成相关的一系列类



（3）调用远程服务

```
public class ClientTest {
    @Test
    public void testClient(){
        JaxWsProxyFactoryBean factory = new JaxWsProxyFactoryBean();
        factory.setAddress("http://localhost:12345/user?wsdl");
        factory.setServiceClass(IUserService.class);
        IUserService ius = (IUserService)factory.create();
        /*String result = ius.sayHi("小陈");
        System.out.println(result);*/
        User user = new User();
        user.setUsername("宋江");
        List<Car> list = ius.findCarsByUser(user);
        System.out.println(list);
    }
}
```

注意：为了更好的测试效果，可以在 Webservice 发布的服务类中添加一些日志信息输出代码。

```
factory.getInInterceptors().add(new LoggingInInterceptor());
factory.getOutInterceptors().add(new LoggingOutInterceptor());
```

为了使得该配置生效，还需要加入 Log4j 的配置文件 log4j.properties

四、CXF 与 Spring 整合(JAX_WS 方式)

1、服务端

开发步骤：

（1）创建 Maven 的 WEB 工程,添加依赖，包括 jaxws 和 spring 的依赖。

```
<dependency>
    <groupId>org.apache.cxf</groupId>
    <artifactId>cxf-rt-frontend-jaxws</artifactId>
    <version>3.0.1</version>
</dependency>
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>4.2.4.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-web</artifactId>
    <version>4.2.4.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-test</artifactId>
    <version>4.2.4.RELEASE</version>
</dependency>
```

（2）修改 web.xml，添加 spring 监听器 ContextLoaderListener，并加载 spring 配置文件

（3）修改 web.xml，添加 CXFServlet

```
<!-- cxf的核心控制器 -->
<servlet>
    <servlet-name>cxf</servlet-name>
    <servlet-class>org.apache.cxf.transport.servlet.CXFServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>cxf</servlet-name>
    <url-pattern>/ws/*</url-pattern>
</servlet-mapping>
```

（4）添加 spring 配置文件,其中 UserServiceImpl 为服务实现类

```
<jaxws:server address="/user">
    <jaxws:serviceBean>
        <bean class="cn.itcast.service.UserServiceImpl"/>
    </jaxws:serviceBean>
</jaxws:server>
```

（5）测试服务端,请求地址：**http://localhost:8080/bj44ws_spring/ws/user?wsdl**

2、客户端

（1）添加客户端的 Spring 配置文件 applicationContext-client.xml，并配置内容如下

```
<!-- 2.实现客户端的配置 -->
<jaxws:client id="userService" address="http://localhost:8080/bj44ws_spring/ws/user?wsdl"
   serviceClass="cn.itcast.service.IUserService"
>
</jaxws:client>
```

（2）加载 spring 配置文件进行 jaxws 与 spring 整合的客户端测试

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("classpath:applicationContext-client.xml")
public class JaxwsClientTest {
    @Autowired
    private IUserService userService;
    @Test
    public void testClient(){
        String result = userService.sayHi("传智.宋江");
        System.out.println("============"+result);
    }
}
```

五.Restful 编程风格

1.REST 起源

REST 这个词，是 Roy Thomas Fielding 在他 2000 年的博士论文中提出的。

Fielding 是一个非常重要的人，他是 HTTP 协议（1.0 版和 1.1 版）的主要设计者、Apache 服务器软件的作者之一、Apache 基金会的第一任主席。所以，他的这篇论文一经发表，就引起了关注，并且立即对互联网开发产生了深远的影响。

他这样介绍论文的写作目的：

"本文研究计算机科学两大前沿----软件和网络----的交叉点。长期以来，软件研究主要关注软件设计的分类、设计方法的演化，很少客观地评估不同的设计选择对系统行为的影响。而相反地，网络研究主要关注系统之间通信行为的细节、如何改进特定通信机制的表现，常常忽视了一个事实，那就是改变应用程序的互动风格比改变互动协议，对整体表现有更大的影响。我这篇文章的写作目的，就是想在符合架构原理的前提下，理解和评估以网络为基础的应用软件的架构设计，得到一个功能强、性能好、适宜通信的架构。"

他对这个互联网软件的架构原则定名为 REST，即 REpresentational State Transfer 的缩写，翻译过来就是表现层状态转移。

REST 描述的是在网络中 client 和 server 的一种交互形式；REST 本身不实用，实用的是如何设计 REST 风格的网络接口

2.Restful 风格的服务

3.Restful 风格优点

基于这种风格架构，软件编写可以更简洁

基于 HTTP 协议， 支持多种消息格式，比如 XML 、JSON

更易于实现缓存机制（第一次访问资源 缓存，第二次访问资源，返回 304 客户端调用本地）

http://localhost:12345/userService/user

POST 请求方式访问 保存操作

PUT 请求方式访问 修改操作

GET 请求方式访问 查询操作

DELETE 请求方式访问 删除操作

原来的方式

http://127.0.0.1/user/queryUser/{id}	GET 方法，根据用户 id 获取数据
http://127.0.0.1/user/updateUser	POST 方法，用户修改
http://127.0.0.1/user/saveUser	POST 方法，用户新增
http://127.0.0.1/user/deleteUser/{id}	GET/POST 方法，用户根据 id 删除

RESTful

http://127.0.0.1/user/{id}	GET 方法，根据用户 id 获取数据
http://127.0.0.1/user/	PUT 方法，用户修改
http://127.0.0.1/user/	POST 方法，用户新增
http://127.0.0.1/user/{id}	DELETE 方法，用户根据 id 删除

http://localhost:9997/userService/user 查询所有用户信息

http://localhost:9997/userService/user/1 查询用户编号为 1 的用户信息

访问服务器资源，采用不同 HTTP 协议请求方式，服务器端可以得知进行 CRUD 哪个操作！

JAX-RS 的 webservice 服务发布方式就是采用了 restful 风格。

六.编写 JAX-RS 服务应用

1.使用 Jax-RS 独立实现服务发布

基于 maven 导入坐标

```
<dependency>
    <groupId>org.apache.cxf</groupId>
    <artifactId>cxf-rt-frontend-jaxrs</artifactId>
    <version>3.0.1</version>
</dependency>
<dependency>
    <groupId>org.apache.cxf</groupId>
    <artifactId>cxf-rt-transports-http-jetty</artifactId>
    <version>3.0.1</version>
</dependency>
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-log4j12</artifactId>
    <version>1.7.12</version>
</dependency>
<dependency>
    <groupId>org.apache.cxf</groupId>
    <artifactId>cxf-rt-rs-client</artifactId>
    <version>3.0.1</version>
```

```
        </dependency>
        <dependency>
            <groupId>org.apache.cxf</groupId>
            <artifactId>cxf-rt-rs-extension-providers</artifactId>
            <version>3.0.1</version>
        </dependency>
        <dependency>
            <groupId>org.codehaus.jettison</groupId>
            <artifactId>jettison</artifactId>
            <version>1.3.7</version>
        </dependency>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.10</version>
            <scope>test</scope>
        </dependency>
```

1、 导入实体类

```
@XmlRootElement(name = "User")
public class User {

@XmlRootElement(name = "Car")
public class Car {
```

@XmlRootElement 指定序列化（转换 XML、JSON） 对象名字

2、编写业务类

```
@Path("/userService")
@Produces("*/*")
public interface IUserService {
    @POST
    @Path("/user")
    @Consumes({"application/xml","application/json"})
    public void saveUser(User user);

    @PUT
    @Path("/user")
    @Consumes({"application/xml","application/json"})
    public void updateUser(User user);

    @GET
    @Path("/user")
    @Produces({ "application/xml", "application/json" })
    public List<User> findAllUsers();

    @GET
    @Path("/user/{id}")
    @Consumes("application/xml")
    @Produces({ "application/xml", "application/json" })
    public User finUserById(@PathParam("id") Integer id);
```

第一种 @Path 服务访问资源路径

如果访问 saveUser 方法 /userService/user

第二种 @Produces 生成（方法返回值） @Consumes 消费 （方法参数）

@Consumes 指定能够处理客户端传递过来数据格式

@Produces 指定能否生成哪种格式数据返回给客户端

第三种 @GET 查询 @PUT 修改 @POST 增加 @DELETE 删除

如果实现某条数据单独查询，使用 url 参数 ！

3、 发布服务

```
public class JaxrsTest {
    public static void main(String[] args) {
        JAXRSServerFactoryBean factory = new JAXRSServerFactoryBean();
        factory.setResourceClasses(User.class,Car.class);
        factory.setAddress("http://localhost:12345/");
        factory.setServiceBean(new UserServiceImpl());

        factory.getInInterceptors().add(new LoggingInInterceptor());
        factory.getInInterceptors().add(new LoggingOutInterceptor());
        factory.create();
    }
}
```

2.编写 JAX-RS 客户端程序

WebClient 工具类使用 （CXF 自带）

```
<dependency>
    <groupId>org.apache.cxf</groupId>
    <artifactId>cxf-rt-rs-client</artifactId>
    <version>3.0.1</version>
</dependency>
```

测试类编写

```
@Test
public void testClient(){
    WebClient client = WebClient.create("http://localhost:12345/userService/user/1");
    client.type(MediaType.APPLICATION_JSON);
    User user = new User();
    user.setId(1);
    user.setUsername("张三");
    user.setCity("北京");
    //client.post(user);
    //client.put(user);
    //client.delete();
}
@Test
public void testClient02(){
    WebClient client = WebClient.create("http://localhost:12345/userService/user");
    //client.type(MediaType.APPLICATION_JSON);
    client.accept(MediaType.APPLICATION_JSON);
    Collection<? extends User> collection = client.getCollection(User.class);

    System.out.println(collection);
}
@Test
```

3.JAX-RS 如何传输 JSON 格式的数据

如果指定客户端要获取 json 内容

错误：Caused by: javax.ws.rs.ProcessingException: No message body writer has been found for class cn.itcast.cxf.domain.User, ContentType: application/json

解决： 在项目引入 json 转换器

```
<dependency>
    <groupId>org.apache.cxf</groupId>
    <artifactId>cxf-rt-rs-extension-providers</artifactId>
    <version>3.0.1</version>
</dependency>
<dependency>
    <groupId>org.codehaus.jettison</groupId>
    <artifactId>jettison</artifactId>
    <version>1.3.7</version>
</dependency>
```

4.JAX-RS 和 Spring 整合开发

1、 建立 maven web 项目并导入 maven 坐标

```
<!-- cxf 进行rs开发 必须导入 -->
<dependency>
    <groupId>org.apache.cxf</groupId>
    <artifactId>cxf-rt-frontend-jaxrs</artifactId>
    <version>3.0.1</version>
</dependency>
<!-- 日志引入 -->
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-log4j12</artifactId>
    <version>1.7.12</version>
</dependency>
<!-- 客户端 -->
<dependency>
    <groupId>org.apache.cxf</groupId>
    <artifactId>cxf-rt-rs-client</artifactId>
    <version>3.0.1</version>
</dependency>
<!-- 扩展json提供者 -->
<dependency>
    <groupId>org.apache.cxf</groupId>
    <artifactId>cxf-rt-rs-extension-providers</artifactId>
    <version>3.0.1</version>
</dependency>
```

```
        <!-- 转换json工具包，被extension providers 依赖 -->
        <dependency>
            <groupId>org.codehaus.jettison</groupId>
            <artifactId>jettison</artifactId>
            <version>1.3.7</version>
        </dependency>
        <!-- spring 核心 -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>4.2.4.RELEASE</version>
        </dependency>
        <!-- spring web集成 -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-web</artifactId>
            <version>4.2.4.RELEASE</version>
        </dependency>
        <!-- spring 整合junit -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-test</artifactId>
            <version>4.2.4.RELEASE</version>
        </dependency>
        <!-- junit 开发包 -->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
        </dependency>
```

在 web.xml 中导入 CXFServlet

```
<servlet>
    <servlet-name>CXFService</servlet-name>
    <servlet-class>org.apache.cxf.transport.servlet.CXFServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>CXFService</servlet-name>
    <url-pattern>/services/*</url-pattern>
</servlet-mapping>
```

2、 导入实体类和 Service

3、 在 spring 配置发布 rs 服务

引入名称空间 xmlns:jaxrs="http://cxf.apache.org/jaxrs"

http://cxf.apache.org/jaxrs http://cxf.apache.org/schemas/jaxrs.xsd

最终访问资源服务路径

服务器根目录地址 + web.xml 配置 + applicationContext.xml address 配置 + 类 @Path + 方法 @Path

```
@Path("/userService")
@Produces("*/*")
public interface IUserService {
```

applicationContext.xml 配置重复

```
<jaxrs:server id="userService" address="/userService" >
    <jaxrs:serviceBeans>
        <bean class="cn.itcast.service.UserServiceImpl" />
    </jaxrs:serviceBeans>
    <jaxrs:inInterceptors>
        <bean class="org.apache.cxf.interceptor.LoggingInInterceptor" />
    </jaxrs:inInterceptors>
    <jaxrs:outInterceptors>
        <bean class="org.apache.cxf.interceptor.LoggingOutInterceptor" />
    </jaxrs:outInterceptors>
</jaxrs:server>
```

4、 编写客户端代码 类似独立服务客户端代码

WebClient 工具实现

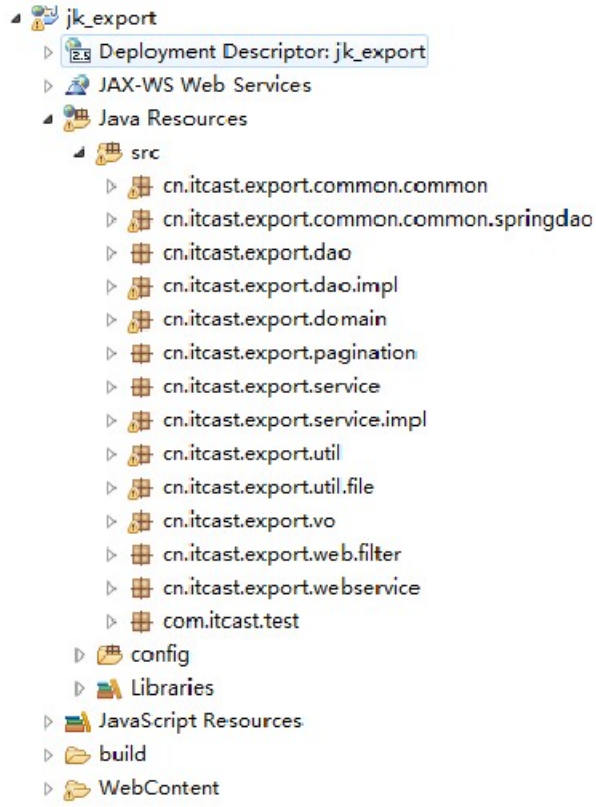
```
@Test
public void testClient(){
     Collection<? extends User> collection = WebClient.create("http://localhost:8080/bj44rs_spring/services/userService/user")
    .accept(MediaType.APPLICATION_JSON).getCollection(User.class);
    System.out.println(collection);
}
@Test
public void testClient02(){
     User user = WebClient.create("http://localhost:8080/bj44rs_spring/services/userService/user/1")
    .accept(MediaType.APPLICATION_JSON).get(User.class);
    System.out.println(user);
}
@Test
public void testClient03(){
     User user = new User();
     user.setId(22);
     user.setCity("北京");
     user.setUsername("传智.宋江");
     WebClient.create("http://localhost:8080/bj44rs_spring/services/userService/user")
    .accept(MediaType.APPLICATION_JSON).post(user);

}
```

七.项目整合应用

1.搭建海关电子报运平台的服务端

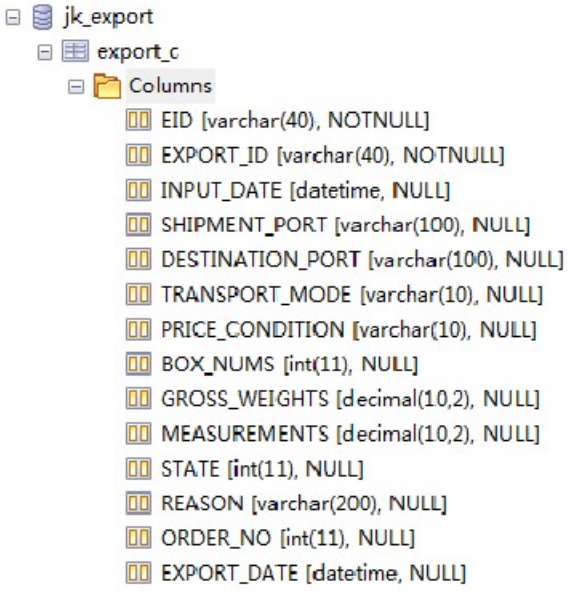
海关报运系统的服务端采用(SSH+MySQL+Restful+JAXRS)实现海关报运平台的 webservice 服务的发布，作为国际物流的商贸公司可以直接调用海关报运平台的服务，从而减少报运专中的工作量，这样我们只要在国际物流云商系统项目中直接调用海关报运平台的 webservice 轻松实现海关报运工作。

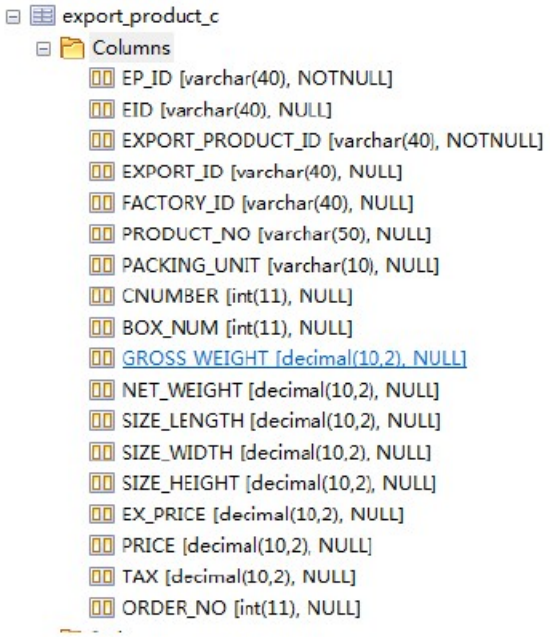


2.搭建海关电子报运系统

开发方式：**SSH+Restful+CXF（JAX-RS）+MySQL** 数据库

3.设计海关报运系统的报运相关的表





4.设计交互的数据结构

本次远程调用的数据结构，采用 XML/JSON 数据方式进行交互

请求海关报运平台的数据结构分析

```
{
 exportId:"",
 inputDate:"",
 shipmentPort:"",
 destinationPort:"",
 transportMode:"",
 priceCondition:"",
 boxNums:"",
 grossWeights:"",
 measurements:"",
 products:[
            {
              exportProductId:"",
              factoryId:"",
              productNo:"",
              packingUnit:"",
              cnumber:"",
              boxNum:"",
              grossWeight:"",
              netWeight:"",
              sizeLength:"",
              sizeWidth:"",
```

```
            sizeHeight:"",
            exPrice:"",
            price:"",
            tax:""
        },
        {
            exportProductId:"",
            factoryId:"",
            productNo:"",
            packingUnit:"",
            cnumber:"",
            boxNum:"",
            grossWeight:"",
            netWeight:"",
            sizeLength:"",
            sizeWidth:"",
            sizeHeight:"",
            exPrice:"",
            price:"",
            tax:""
        }
    ]
}
```

在 JAX-RS 下可以采用实体类来实现数据的交互,我们可以为系统设计专用的 VO，专门用于实现数据交互。

```
import java.io.Serializable;
@XmlRootElement(name="export")
public class ExportVo extends Export {


}
```

```
@XmlRootElement(name="exportProduct")
public class ExportProductVo extends ExportProduct {


}
```

海关报运平台响应结果分析

状态字段：响应为 2 代表通过　响应为 3 代表不通过（此时会带上一个 REASON 代表不通过的原因）

真正对应的数据：

```
{
 exportId:"",
 state:"",
 remark:"",
 products:[ {
            exportProductId:"",
```

```
            tax:""
        },
        {
            exportProductId:"",
            tax:""
        }]
}
```

在 JAX-RS 下，也可以通过专门的实体对象用于封装 JSON 数据实现响应结果的处理。

```
@XmlRootElement(name="export")
public class ExportResult {
    private String exportId;
    private Integer state;
    private String remark;
    private Set<ExportProductResult> products;

@XmlRootElement(name="products")
public class ExportProductResult {
    private String exportProductId;
    private Double tax;
```

八．作业

1.使用 JAX-WS 实现 WebService 服务的发布与调用
2.使用 JAX-WS 与 Spring 整合实现 WebService 服务的发布与调用
3.理解 Restful 编程思想
4.使用 JAX-RS 实现 WebService 服务的发布与调用
5.实现 JAX-RS 与 Spring 整合实现 WebService 服务的发布与调用
6.熟悉《海关报运平台》基于 Restful 风格的 JAX-RS WebService 服务发布