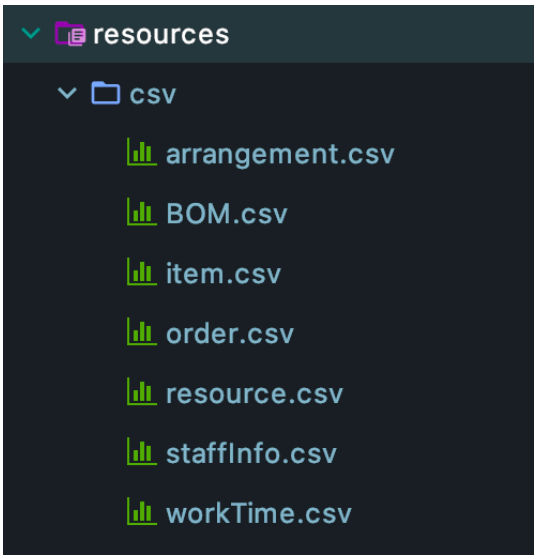
说明文档

一. 数据源说明

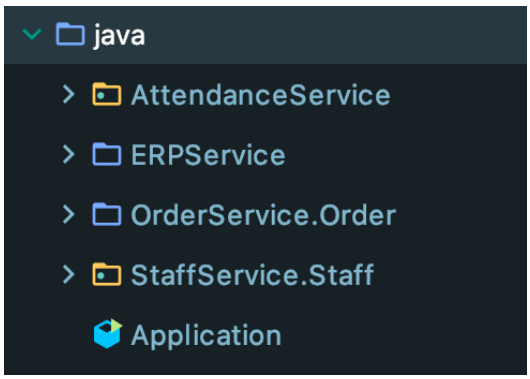
我们将提供的基础数据（2）表提取有效信息形成csv文件，日历、班次、资源、物品信息、产品BOM、订单信息分别对应arrangement.csv、workTime.csv、resource.csv、item.csv、BOM.csv、order.csv，staffInfo.csv用于人事系统中的身份验证。



二. 服务端代码说明

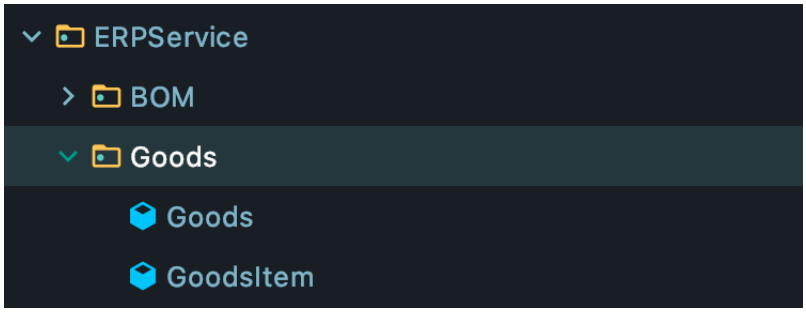
1. 目录结构

AttendanceService对应考勤系统，ERPService对应ERP系统，OrderService对应订单管理系统，StaffService对应人事系统。



2. 代码说明

以ERP系统中的获取物品信息该Web Service为例



GoodsItem.java是Goods.java所需要的持久化对象，如下图所示

```
public GoodsItem(String code, String name, String property, String measurementUnit, String color, String length, String hardness, String otherSpecification, String
    this.code = code;
    this.name = name;
    this.property = property;
    this.measurementUnit = measurementUnit;
    this.color = color;
    this.length = length;
    this.hardness = hardness;
    this.otherSpecification = otherSpecification;
    this.materialPreparation = materialPreparation;
    this.minimumPackingQuantity = minimumPackingQuantity;
}

public void show() {
    System.out.println("物料编码: " + code + "\t物料描述: " + name + "\t物品属性: " + property + "\t计量单位: " + measurementUnit + "\t颜色: " + color + "\t长度: " + ler
}

public String getCode() { return code; }

public void setCode(String code) { this.code = code; }
```

Goods.java是用来完成ERP系统的获取物品信息的Webservice逻辑代码，由于使用csv文件作为后台数据模拟数据存储，需要从本地读取相应的csv文件（此处是item.csv）来获得想要的信息，getAllGoods方法是获取所有物品信息的Webservice。

```
@WebService(
        name = "Goods",
        endpointInterface = "ERPService.Goods.Goods"
)
public class Goods {
    private ArrayList<GoodsItem> readCSV() {
        ArrayList<GoodsItem> goodsItems = new ArrayList<>();

        try{
            BufferedReader reader = new BufferedReader(new InputStreamReader(new FileInputStream( name: "src/main/resources/csv/item.csv"),  charsetName: "gbk"));
            reader.readLine();
            String line = null;
            while((line = reader.readLine()) != null) {
```

```
@WebMethod
public ArrayList<GoodsItem> getAllGoods() {
    ArrayList<GoodsItem> goodsItems = readCSV();
    return goodsItems;
}
```

在Application中发布服务

```
String address1 = "http://localhost:8080/goodsService";
Endpoint.publish(address1, new Goods());
System.out.println("GoodsService Published Successfully!");
System.out.println("Address1:" + address1);
```

运行客户端代码后，服务成功启动，在浏览器输入http://localhost:8080/goodsService?wsdl即可看到生成的WSDL文件了

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_221.jdk/Contents/Home/bin/java ..
ResourceService Published Successfully!
Address:http://localhost:8080/resourceService
GoodsService Published Successfully!
Address1:http://localhost:8080/goodsService
BOMService Published Successfully!
Address2:http://localhost:8080/BOMService
StaffService Published Successfully!
Address3:http://localhost:8080/StaffService
OrderService Published Successfully!
Address4:http://localhost:8080/OrderService
```

This XML file does not appear to have any style information associated with it. The document tree is shown below.


```
<!--
    Published by JAX-WS RI (http://jax-ws.java.net). RI's version is JAX-WS RI 2.2.9-b130926.1035 svn-revision#5f6196f2b90e9460065a4c2f4e30e065b245e51e
-->
<!--
    Generated by JAX-WS RI (http://jax-ws.java.net). RI's version is JAX-WS RI 2.2.9-b130926.1035 svn-revision#5f6196f2b90e9460065a4c2f4e30e065b245e51e
-->
<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://Goods.ERPService/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://Goods.ERPService/" name="GoodsService">
  <types>
    <xsd:schema>
      <xsd:import namespace="http://Goods.ERPService/" schemaLocation="http://localhost:8080/goodsService?xsd=1"/>
    </xsd:schema>
  </types>
  <message name="getAllGoods">
    <part name="parameters" element="tns:getAllGoods"/>
  </message>
  <message name="getAllGoodsResponse">
    <part name="parameters" element="tns:getAllGoodsResponse"/>
  </message>
  <portType name="Goods">
    <operation name="getAllGoods">
      <input wsam:Action="http://Goods.ERPService/Goods/getAllGoodsRequest" message="tns:getAllGoods"/>
      <output wsam:Action="http://Goods.ERPService/Goods/getAllGoodsResponse" message="tns:getAllGoodsResponse"/>
    </operation>
  </portType>
  <binding name="GoodsPortBinding" type="tns:Goods">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    <operation name="getAllGoods">
      <soap:operation soapAction=""/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
  </binding>
  <service name="GoodsService">
    <port name="GoodsPort" binding="tns:GoodsPortBinding">
      <soap:address location="http://localhost:8080/goodsService"/>
    </port>
  </service>
</definitions>
```


三. 客户端代码说明

1. 使用JAX-WS生成客户端代码

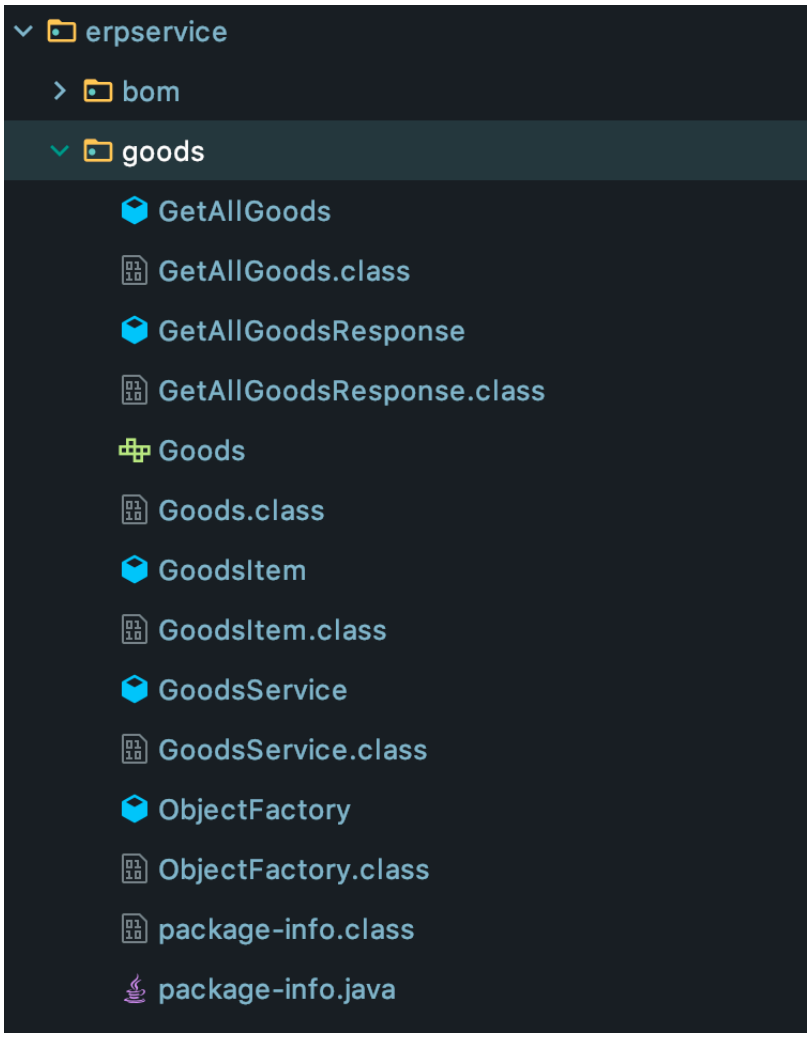
在终端输入以下指令

```
wsimport wsdl地址 -keep -d 保存位置路径
```

以goodService为例，输入以下指令即可在指定位置获得客户端代码，再将客户端代码移动至Client项目的src/main/java下即可

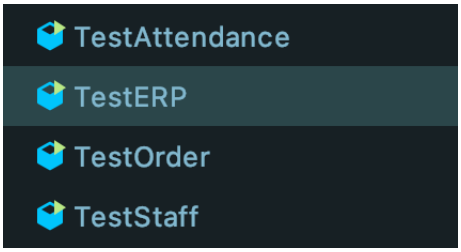
```
wsimport http://localhost:8080/goodsService?wsdl -keep -d
/Users/zhaoxingrui/Desktop
```

生成的客户端代码目录结构如下



2. 测试客户端

编写四个测试代码文件对客户端代码进行测试，比如测试goodService（位于TestERP.java文件中），测试代码如下，启动服务端后，打印所有物品的code和name信息




```java
GoodsService goodsService = new GoodsService();
Goods goodsPort = goodsService.getGoodsPort();
List<GoodsItem> goodsItems = goodsPort.getAllGoods();
for (GoodsItem goodsItem: goodsItems){
    System.out.println(goodsItem.getCode() + " " + goodsItem.getName());
}
```

测试结果（部分）如下图所示