面向服务的体系结构实验报告

实验名称：	创建 SOAP Web Services
实验日期：	2023-10-08
实验地点：	文宣楼 B311
提交日期：	2023-10-09

学号：	33920212204567
姓名：	任宇
专业年级：	软工 2021 级
学年学期：	2023-2024 学年第一学期

1.实验环境

- Windows 10（64 位）
- Oracle SOA Suite 12.2.1.4.0(64 位)

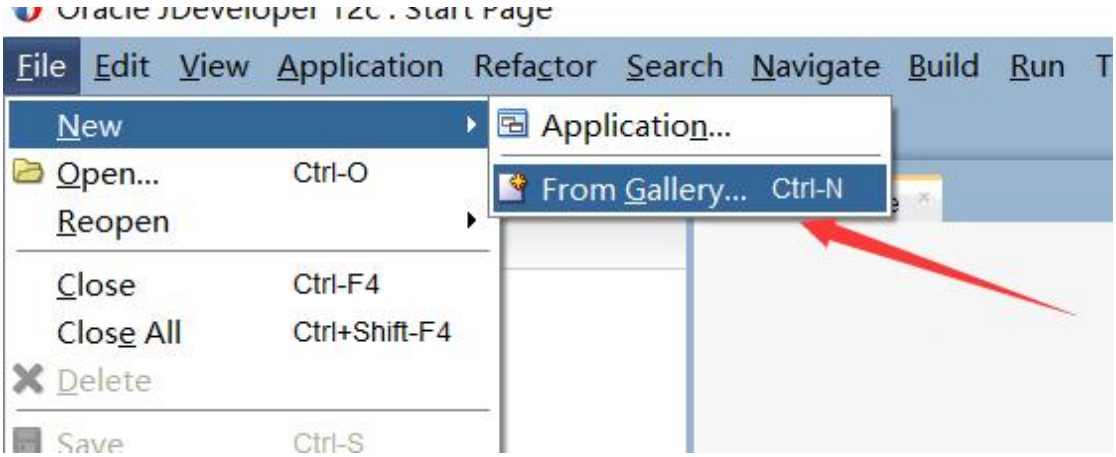
2.实验目的

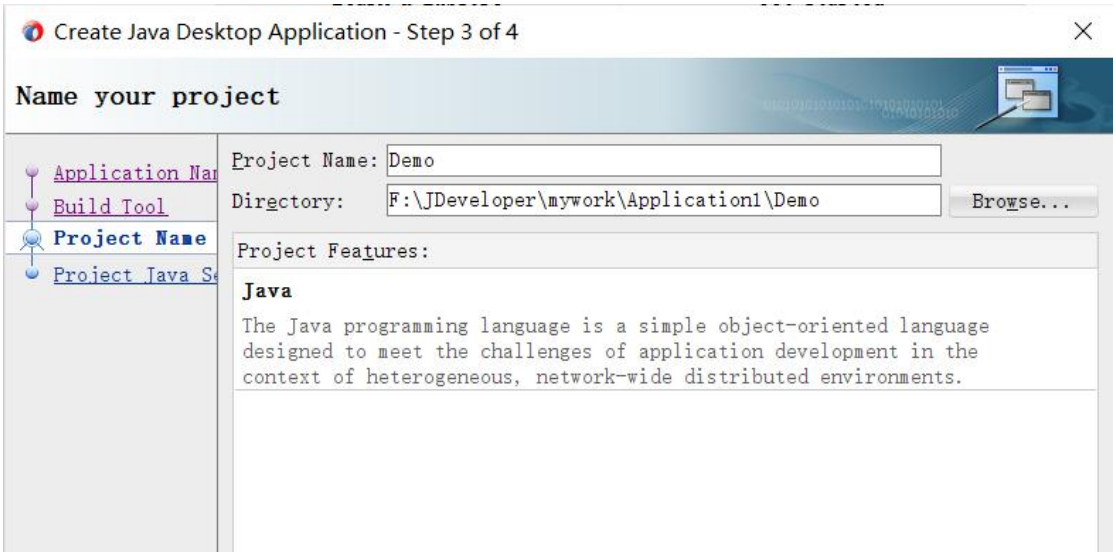
- 掌握创建 SOAP web service 的方法。

3.实验内容和步骤

1）创建一个 Plain Old Java Object（POJO）对象：

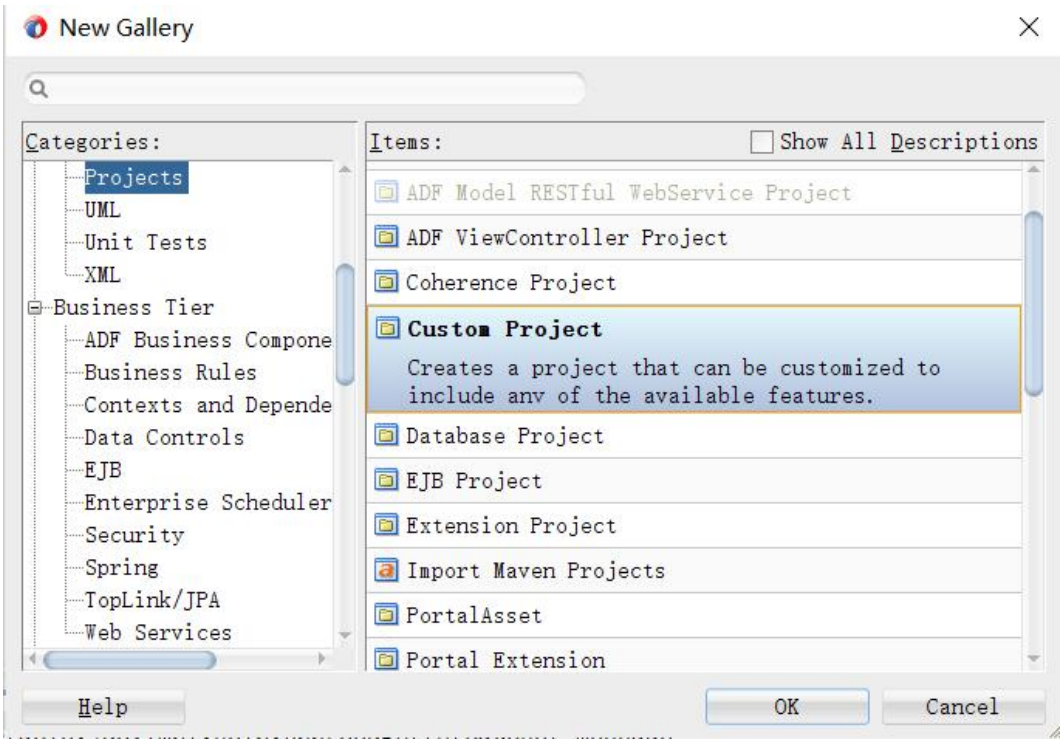
创建应用：

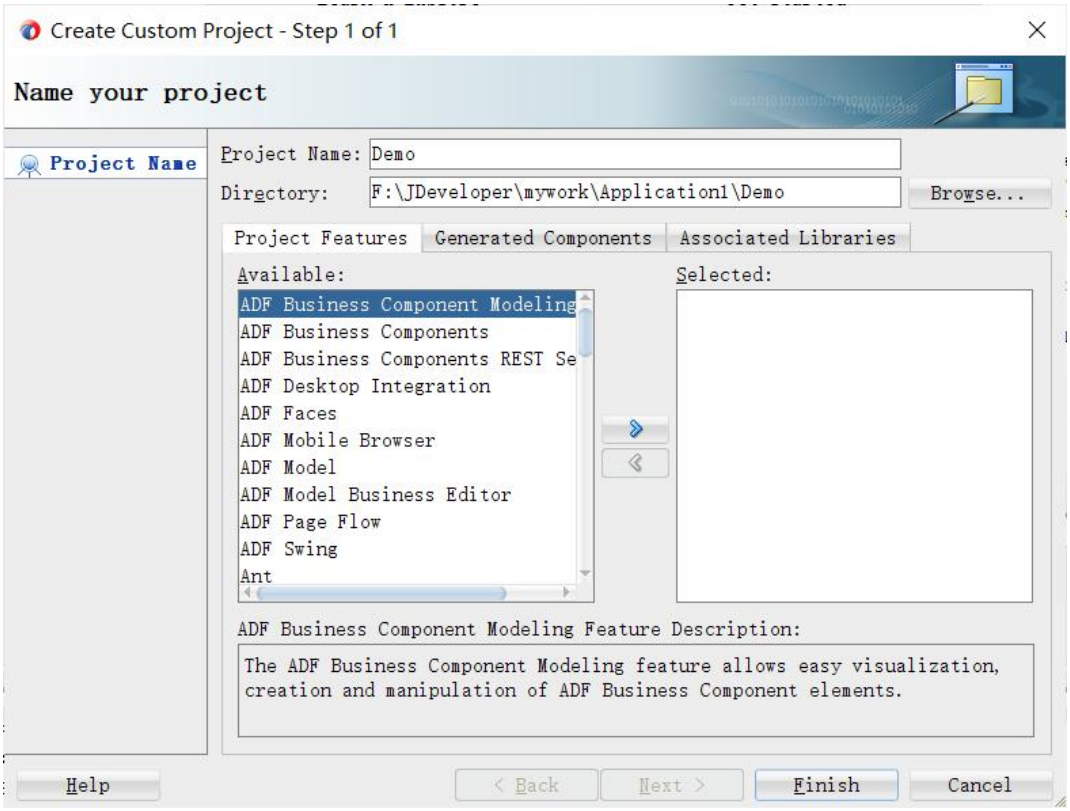




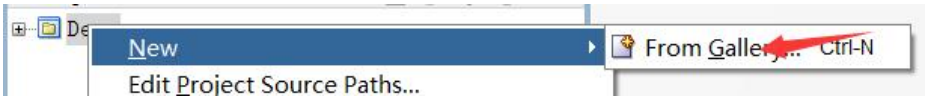
创建新项目：

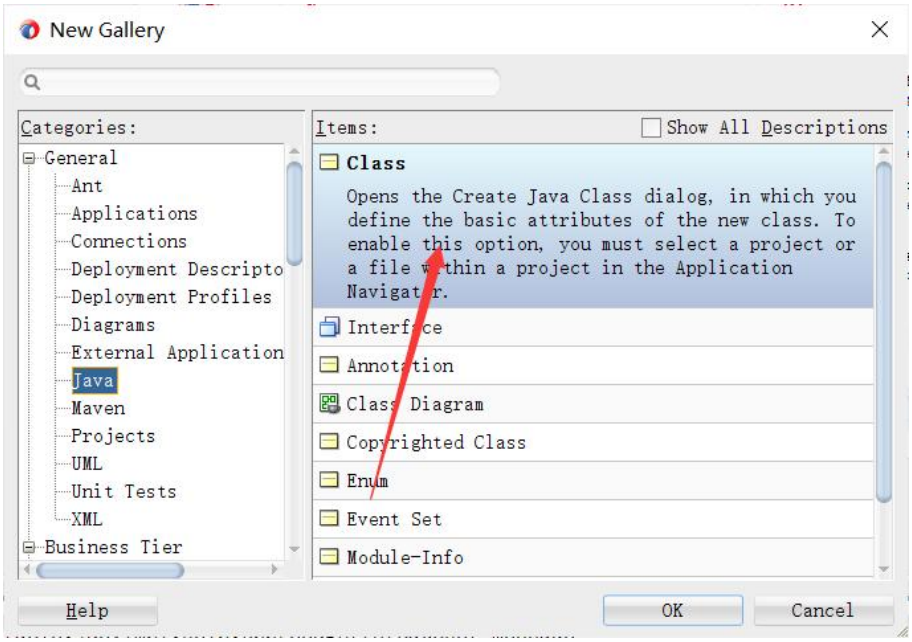

Applications
Application Servers
Start Page
Application1
Project
Applica
Data Co
Recent
New
Open Project...
Close Application
Application...
Project...
From Gallery... Ctrl-N


New Gallery
Categories:
Items:
Show All Descriptions
Projects
UML
Unit Tests
XML
Business Tier
ADF Business Compone
Business Rules
Contexts and Depende
Data Controls
EJB
Enterprise Scheduler
Security
Spring
TopLink/JPA
Web Services
ADF Model RESTful WebService Project
ADF ViewController Project
Coherence Project
Custom Project
Creates a project that can be customized to include anv of the available features.
Database Project
EJB Project
Extension Project
Import Maven Projects
PortalAsset
Portal Extension
Help
OK
Cancel


Create Custom Project - Step 1 of 1
Name your project
Project Name
Project Name: Demo
Directory: F:\JDeveloper\mywork\Application1\Demo
Browse...
Project Features
Generated Components
Associated Libraries
Available:
ADF Business Component Modeling
ADF Business Components
ADF Business Components REST Se
ADF Desktop Integration
ADF Faces
ADF Mobile Browser
ADF Model
ADF Model Business Editor
ADF Page Flow
ADF Swing
Ant
Selected:
ADF Business Component Modeling Feature Description:
The ADF Business Component Modeling feature allows easy visualization, creation and manipulation of ADF Business Component elements.
Help
< Back
Next >
Finish
Cancel

创建 Java 类：


New
From Galler...
Ctrl-N
Edit Project Source Paths...

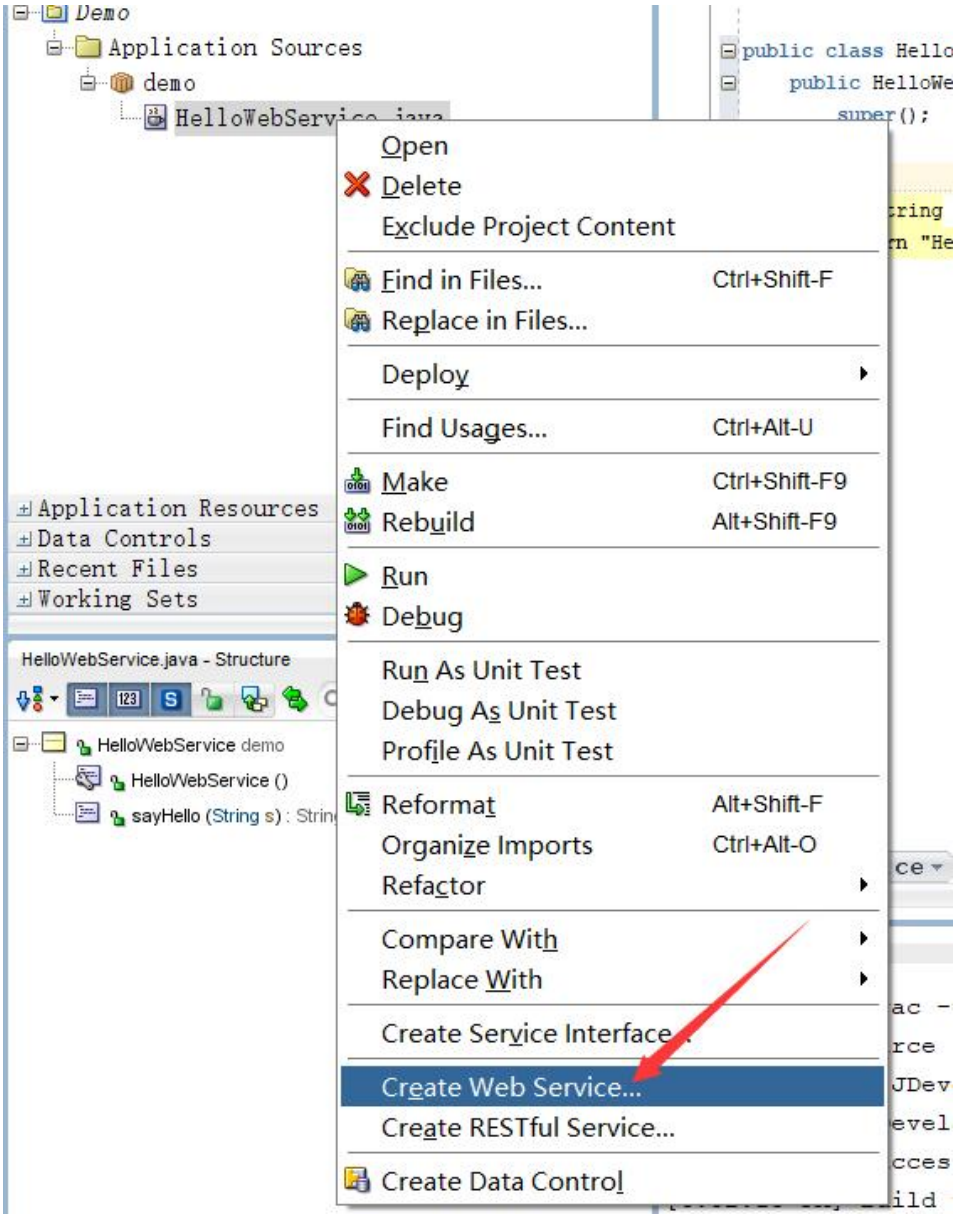

New Gallery
Categories:
General
Ant
Applications
Connections
Deployment Descripto
Deployment Profiles
Diagrams
External Application
Java
Maven
Projects
UML
Unit Tests
XML
Business Tier
Items:
Show All Descriptions
Class
Opens the Create Java Class dialog, in which you define the basic attributes of the new class. To enable this option, you must select a project or a file within a project in the Application Navigator.
Interface
Annotation
Class Diagram
Copyrighted Class
Enum
Event Set
Module-Info
Help
OK
Cancel


Create Java Class
Enter the details of your new class.
Name: HelloWebService
Package: demo
Extends: java.lang.Object
Optional Attributes
Implements:
Access Modifiers
public
package private
Other Modifiers
<None>
abstract
final
Constructors from Superclass
Implement Abstract Methods
Main Method
Messages:
Help
OK
Cancel

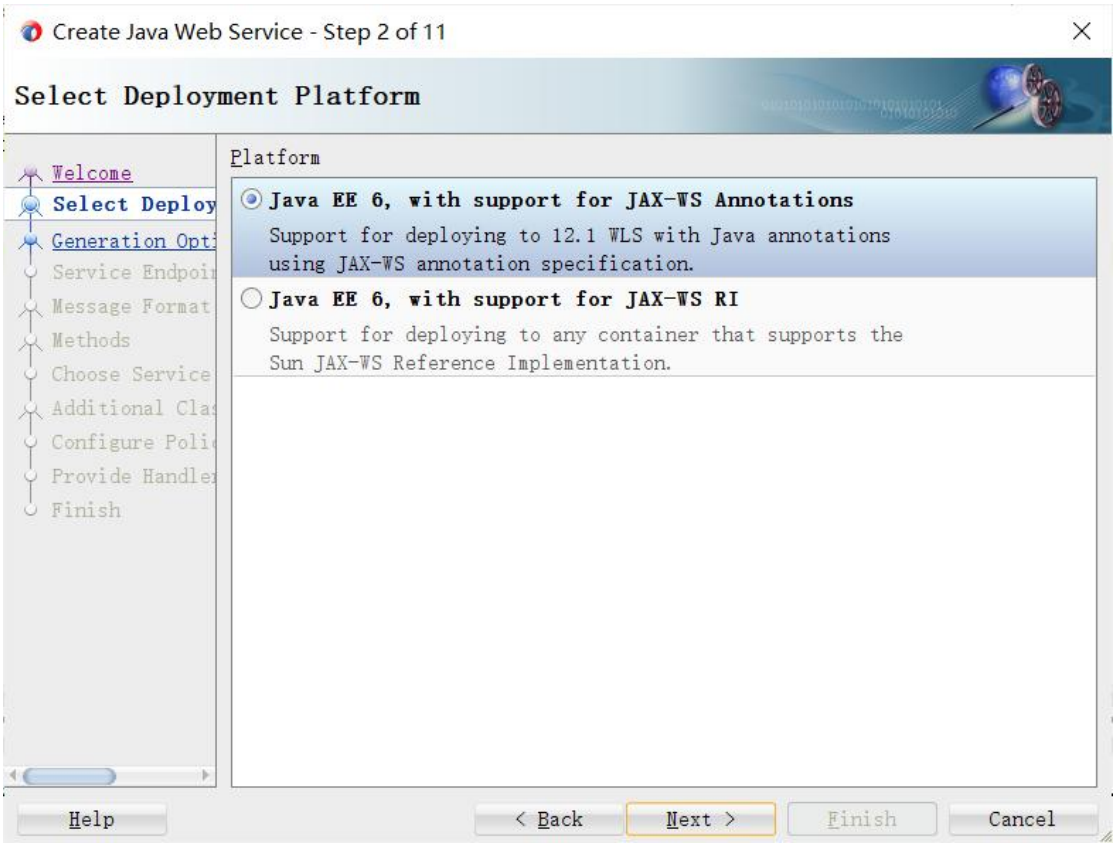
编写方法：

```
package demo;

public class HelloWebService {
    public HelloWebService() {
        super();
    }

    public String sayHello (String s){
        return "Hello "+s;
    }

}
```

2) 创建 web service 类：



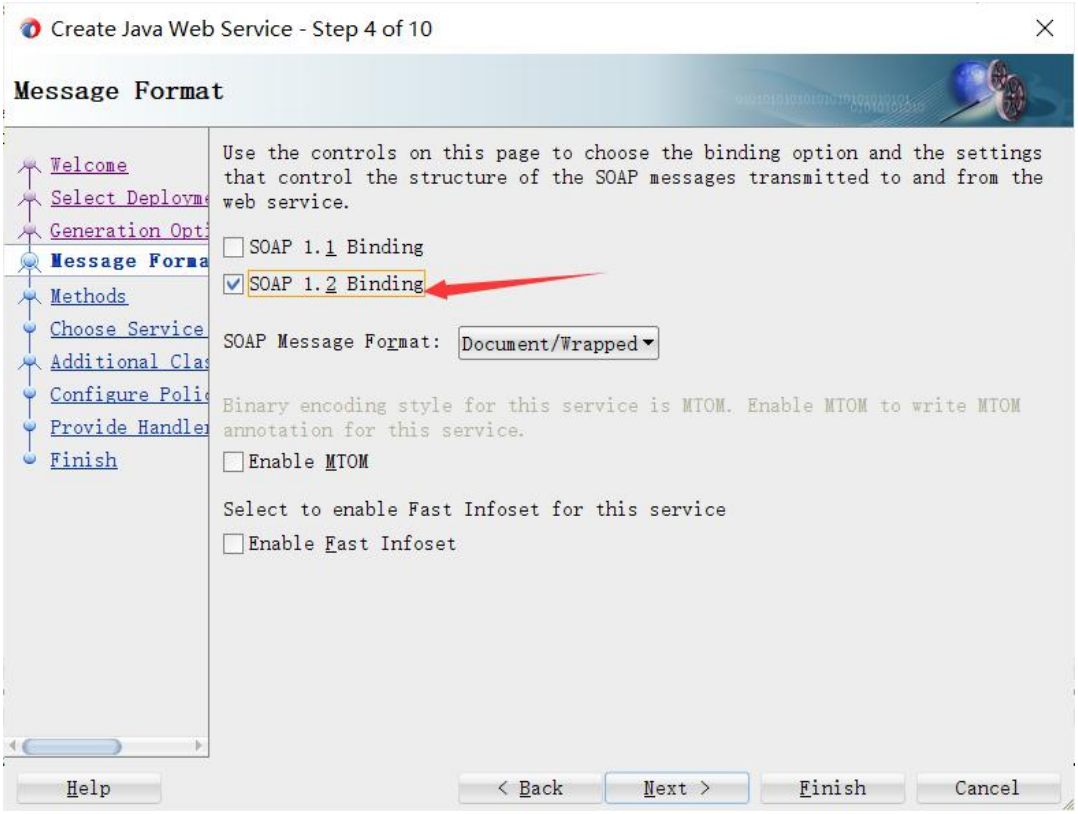
选择部署平台：


Create Java Web Service - Step 2 of 11
Select Deployment Platform
Welcome
Select Deploy
Generation Opti
Service Endpoin
Message Format
Methods
Choose Service
Additional Clas
Configure Polic
Provide Handle
Finish
Platform
Java EE 6, with support for JAX-WS Annotations
Support for deploying to 12.1 WLS with Java annotations using JAX-WS annotation specification.
Java EE 6, with support for JAX-WS RI
Support for deploying to any container that supports the Sun JAX-WS Reference Implementation.
Help
< Back
Next >
Finish
Cancel

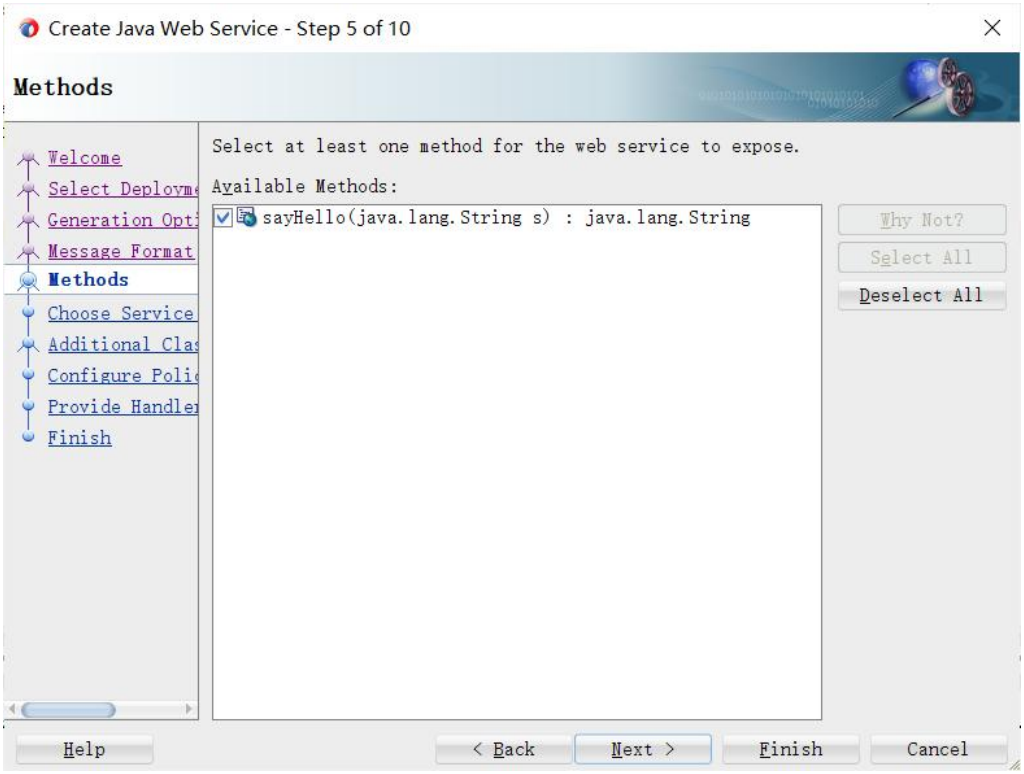
生成选项：


Create Java Web Service - Step 3 of 11
Generation Options
Welcome
Select Deployme
Generation Op
Service Endpoin
Message Format
Methods
Choose Service
Additional Clas
Configure Polic
Provide Handle
Finish
Select or type the Java class or Stateless Session EJB you want to publish (Note: If you type a name for which no class exists, you will be offered the option to have the wizard generate a default implementation class with that name).
Component To Publish:
demo.HelloWebService
Web Service Name:
My_HelloWebService
Port Name
My_HelloWebServicePort
Add SEI
Help
< Back
Next >
Finish
Cancel

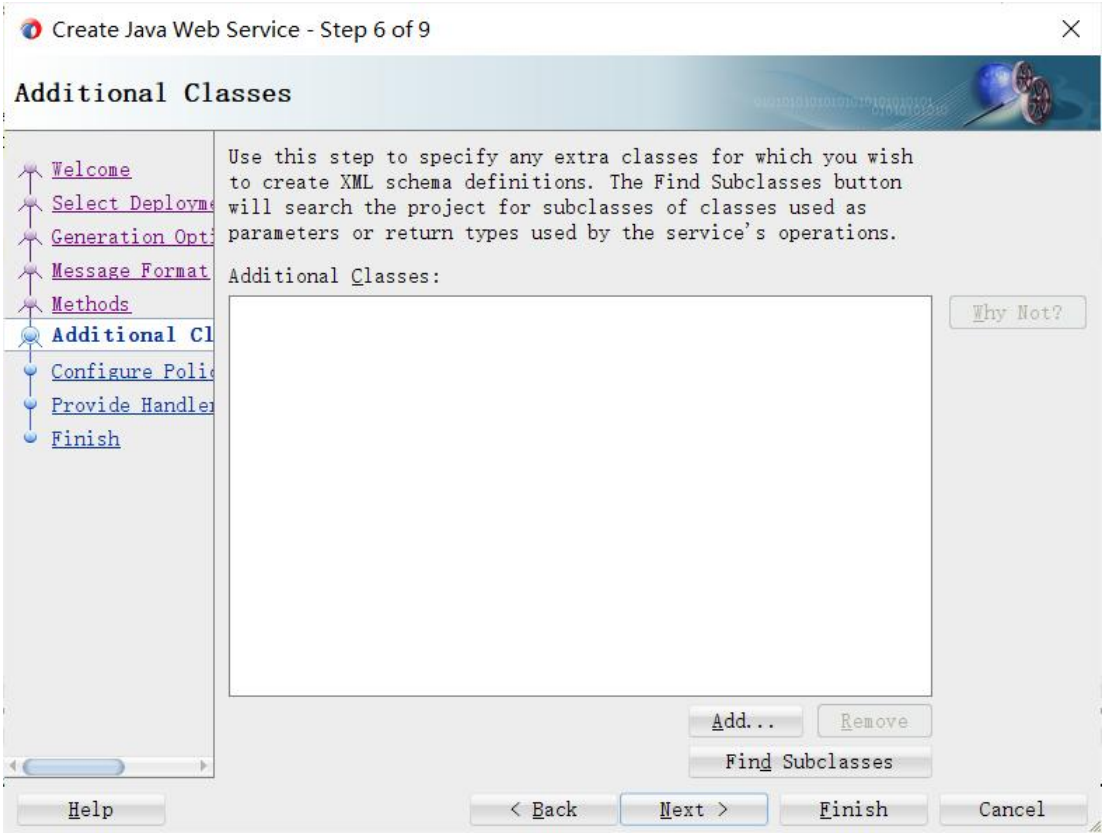
信息格式：


Create Java Web Service - Step 4 of 10
Message Format
Welcome
Select Deployme
Generation Opti
Message Forma
Methods
Choose Service
Additional Clas
Configure Polic
Provide Handler
Finish
Use the controls on this page to choose the binding option and the settings that control the structure of the SOAP messages transmitted to and from the web service.
SOAP 1.1 Binding
SOAP 1.2 Binding
SOAP Message Format:
Document/Wrapped
Binary encoding style for this service is MTOM. Enable MTOM to write MTOM annotation for this service.
Enable MTOM
Select to enable Fast Infoset for this service
Enable Fast Infoset
Help
< Back
Next >
Finish
Cancel

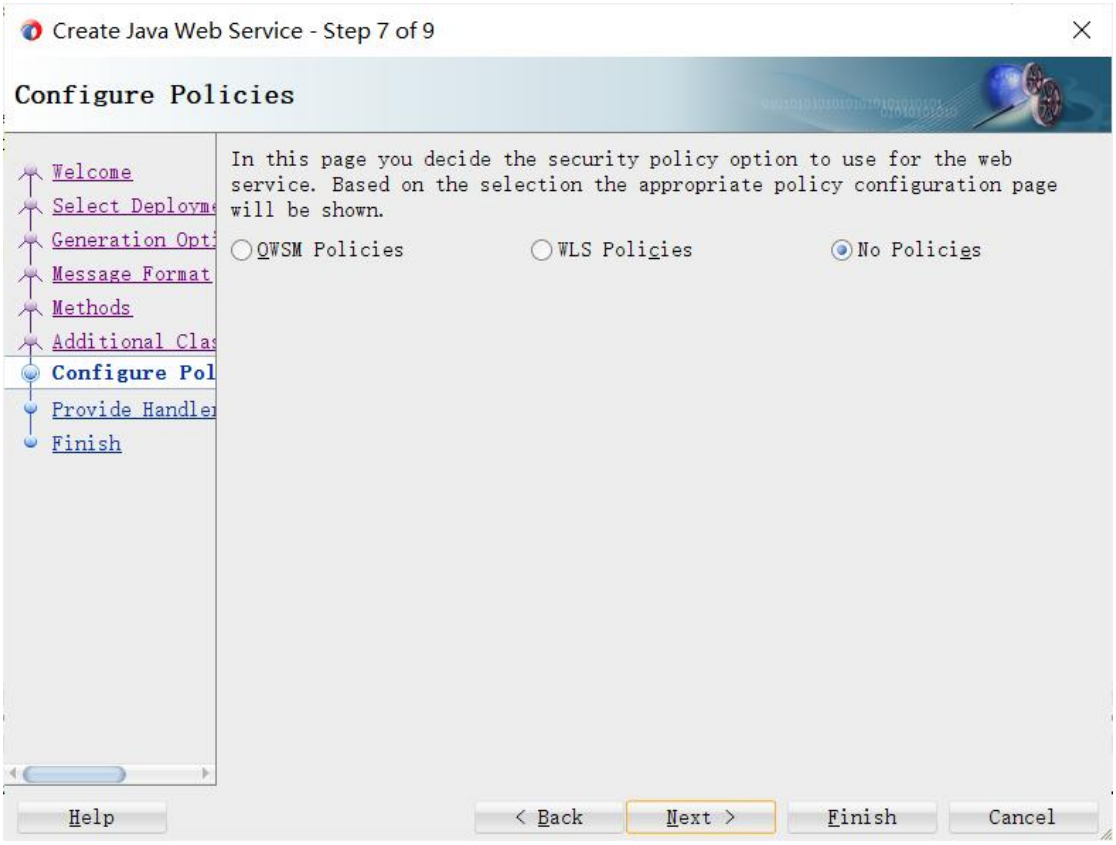
方法：

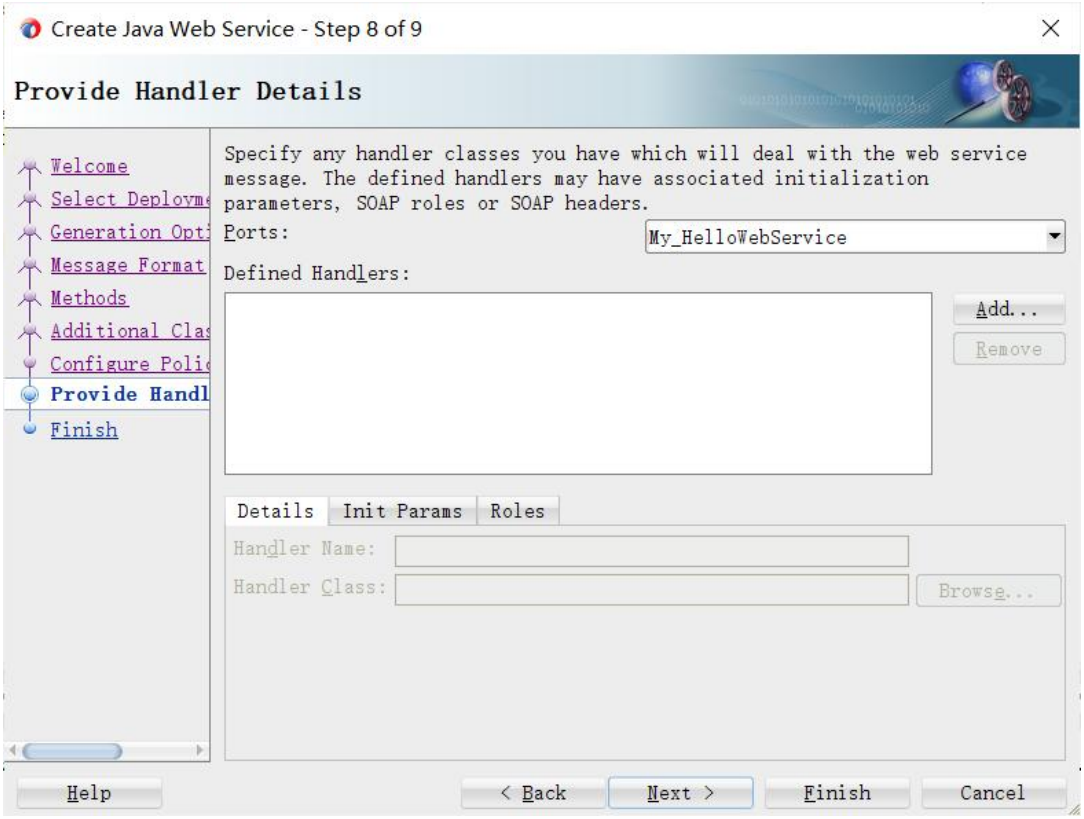

Create Java Web Service - Step 5 of 10
Methods
Welcome
Select Deployme
Generation Opti
Message Format
Methods
Choose Service
Additional Clas
Configure Polic
Provide Handler
Finish
Select at least one method for the web service to expose.
Available Methods:
sayHello(java.lang.String s) : java.lang.String
Why Not?
Select All
Deselect All
Help
< Back
Next >
Finish
Cancel

其他类：

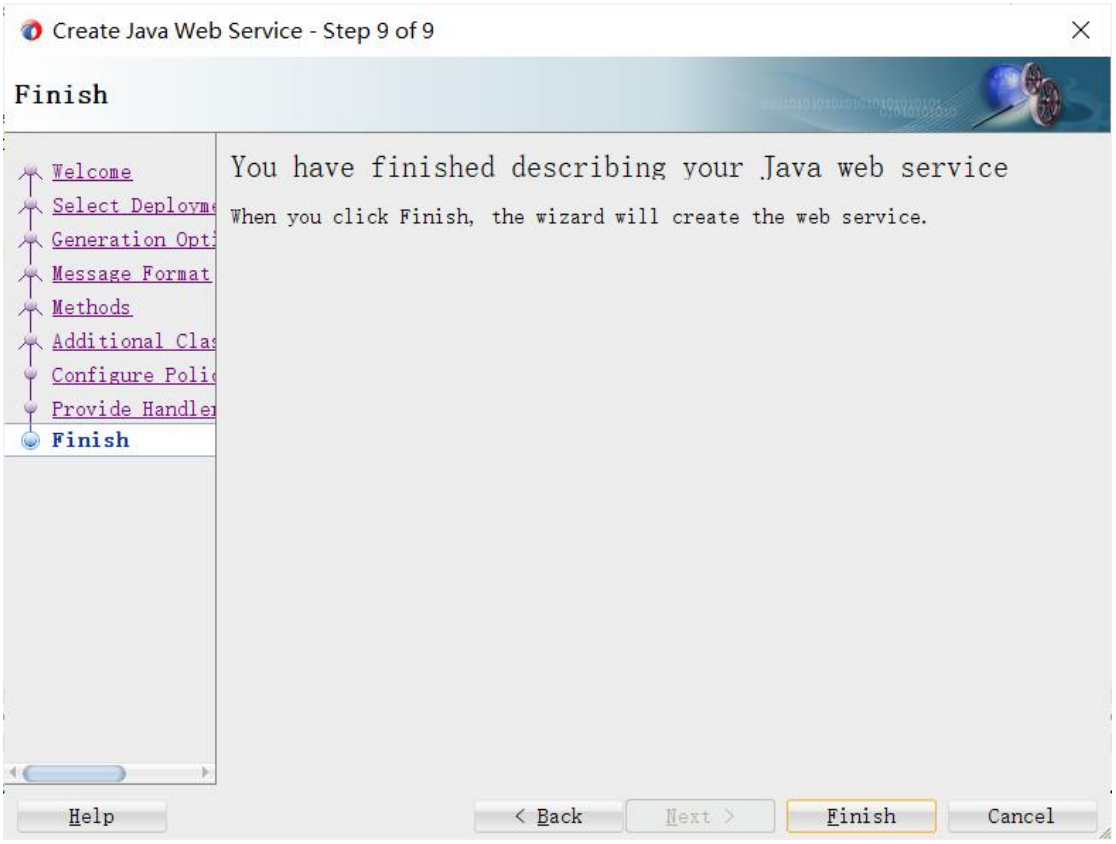

Create Java Web Service - Step 6 of 9
Additional Classes
Welcome
Select Deploym
Generation Opti
Message Format
Methods
Additional Cl
Configure Polic
Provide Handle
Finish
Use this step to specify any extra classes for which you wish to create XML schema definitions. The Find Subclasses button will search the project for subclasses of classes used as parameters or return types used by the service's operations.
Additional Classes:
Why Not?
Add...
Remove
Find Subclasses
Help
< Back
Next >
Finish
Cancel

配置政策：


Create Java Web Service - Step 7 of 9
Configure Policies
Welcome
Select Deploym
Generation Opti
Message Format
Methods
Additional Clas
Configure Pol
Provide Handle
Finish
In this page you decide the security policy option to use for the web service. Based on the selection the appropriate policy configuration page will be shown.
OWSM Policies
WLS Policies
No Policies
Help
< Back
Next >
Finish
Cancel


Create Java Web Service - Step 8 of 9
Provide Handler Details
Welcome
Select Deployme
Generation Opti
Message Format
Methods
Additional Clas
Configure Polic
Provide Handl
Finish
Specify any handler classes you have which will deal with the web service message. The defined handlers may have associated initialization parameters, SOAP roles or SOAP headers.
Ports:
My_HelloWebService
Defined Handlers:
Add...
Remove
Details
Init Params
Roles
Handler Name:
Handler Class:
Browse...
Help
< Back
Next >
Finish
Cancel

结束：

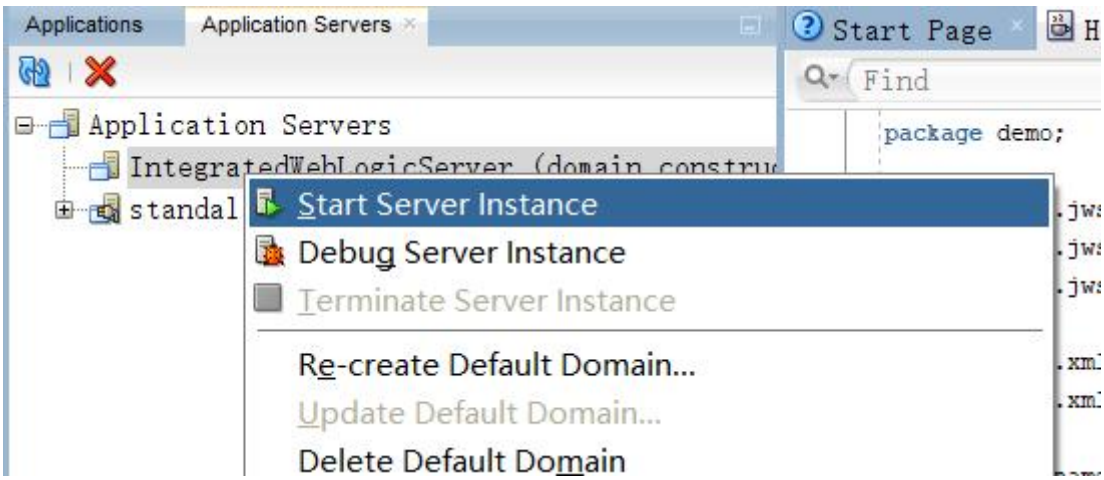

Create Java Web Service - Step 9 of 9
Finish
Welcome
Select Deploym
Generation Opti
Message Format
Methods
Additional Clas
Configure Polic
Provide Handle
Finish
You have finished describing your Java web service
When you click Finish, the wizard will create the web service.
Help
< Back
Next >
Finish
Cancel

完成后可见原 java 类中添加了新的注释 annotations：@WebService、@BindingType、@WebMethod 和 @WebParam：

```
package demo;

import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;

import javax.xml.ws.BindingType;
import javax.xml.ws.soap.SOAPBinding;

@WebService(name = "My_HelloWebService", serviceName = "My_HelloWebService", portName = "My_HelloWebServicePort")
@BindingType(SOAPBinding.SOAP12HTTP_BINDING)
public class HelloWebService {
    public HelloWebService() {
        super();
    }

    @WebMethod
    public String sayHello (@WebParam(name = "arg0") String s){
        return "Hello "+s;
    }

}
```

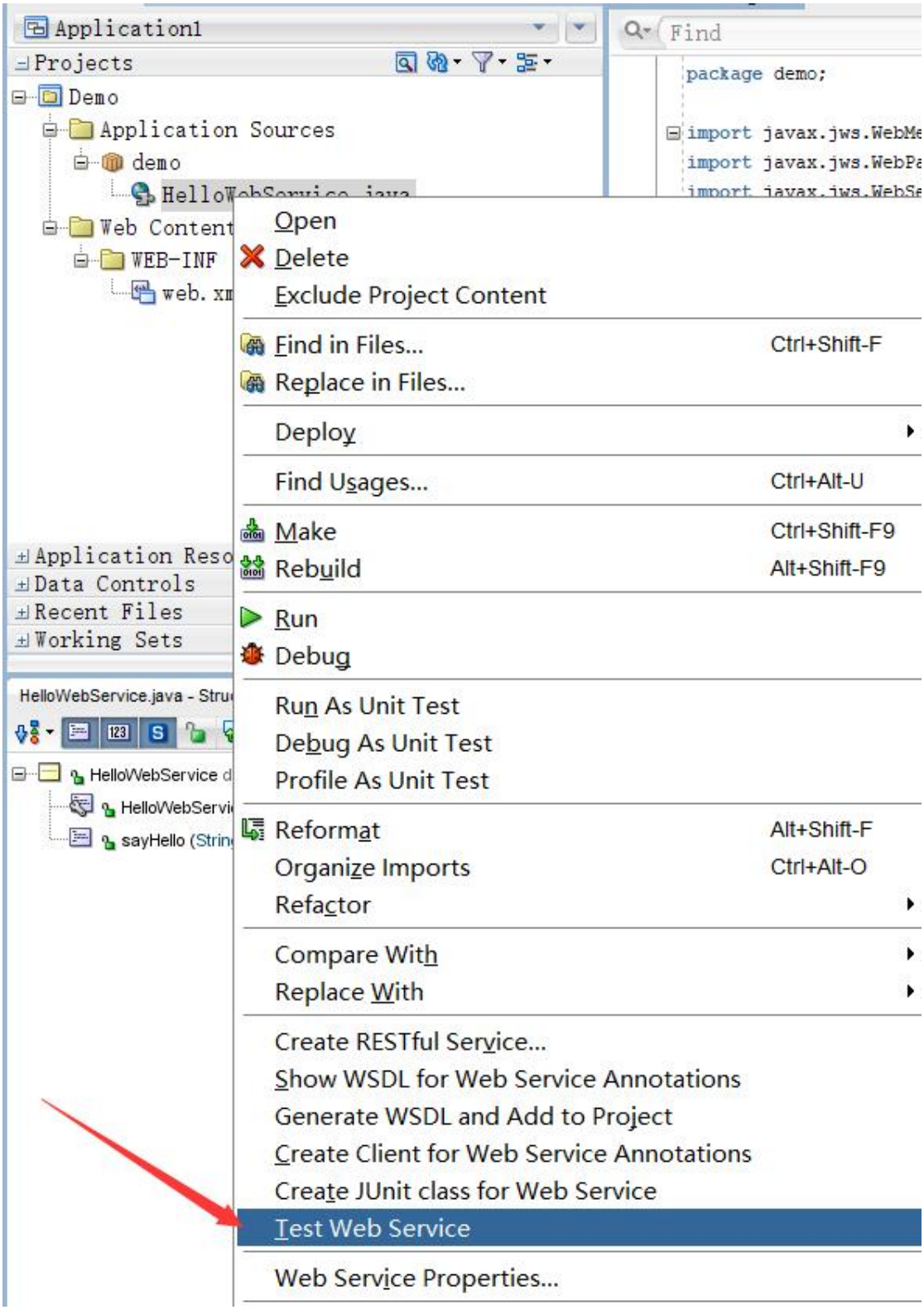
3) 测试 web service：

启动默认域：

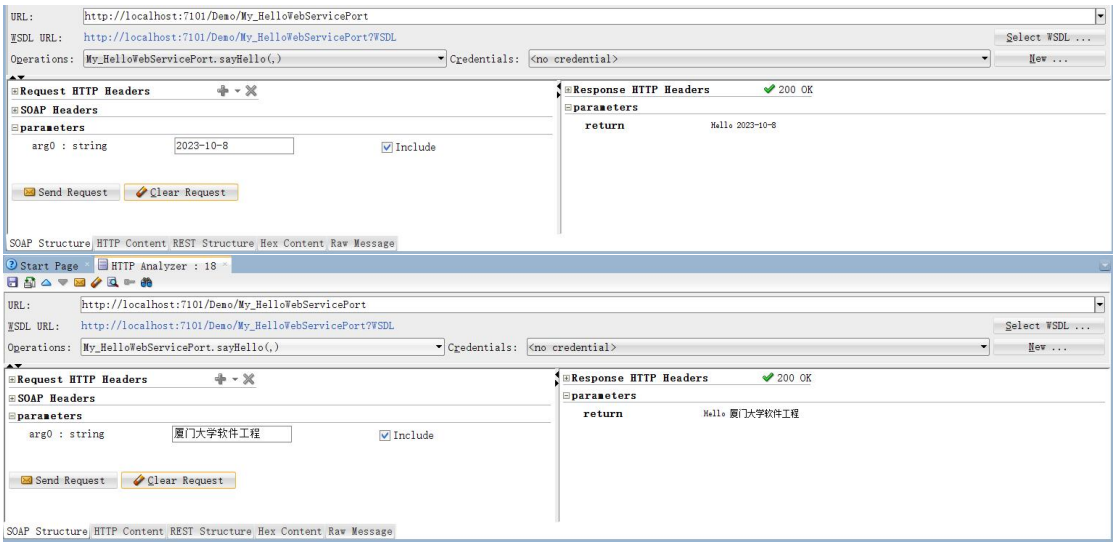



```
[07:26:46 PM] IntegratedWebLogicServer Domain is invalid.  Regenerating it...
[07:26:47 PM] Creating IntegratedWebLogicServer Domain...
[07:27:58 PM] Extending IntegratedWebLogicServer Domain...
[07:29:09 PM] Extending IntegratedWebLogicServer Domain...
[07:29:56 PM] Extending IntegratedWebLogicServer Domain...
[07:31:16 PM] Extending IntegratedWebLogicServer Domain...
[07:32:22 PM] IntegratedWebLogicServer Domain processing completed successfully.
```

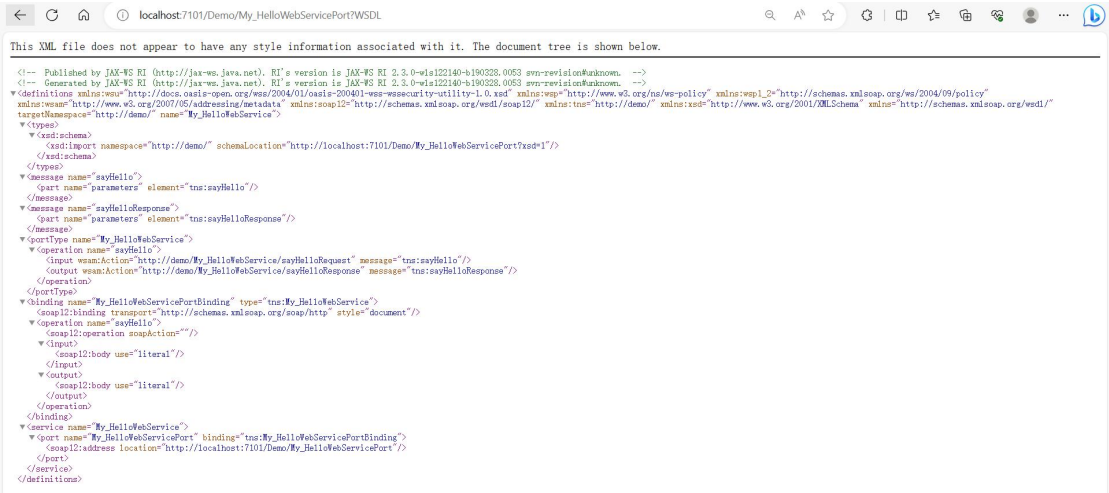
测试 web service：



在 HTTP Analyzer 编辑器中测试，可见成功运行：



4) 分析 WSDL 文档：



总的来说，这个 WSDL 文档描述了一个名为“My_HelloWebService”的 Web 服务，该服务有一个名为“sayHello”的操作，接受一个消息并返回一个响应消息。分析各部分内容：

a) 文档注释:

```
<!-- Published by JAX-WS RI (http://jax-ws.java.net). RI's version is JAX-WS RI 2.3.0-wls122140-b190328.0053 svn-revision#unknown. -->
<!-- Generated by JAX-WS RI (http://jax-ws.java.net). RI's version is JAX-WS RI 2.3.0-wls122140-b190328.0053 svn-revision#unknown. -->
```

由 JAX-WS RI 发布和生成，提供了用于生成该 WSDL 的工具版本信息。

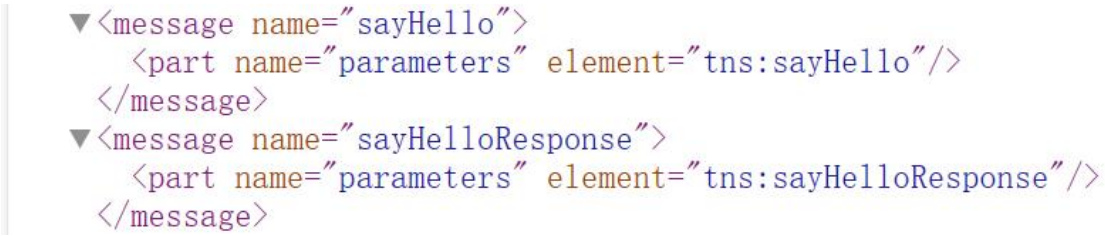
b) 定义与命名空间:

这些命名空间提供了 XML 文档中使用的各种元素和属性的上下文。

c) 类型定义:

xsd:schema 定义了 Web 服务使用的数据类型。在此文档中，它从外部位置 http://localhost:7101/Demo/My_HelloWebServicePort?xsd=1 导入了一个 schema，该 schema 位于 http://demo/命名空间。

d) 消息定义:



sayHello：描述了一个名为"sayHello"的消息，包含一个名为"parameters"的部分，该部分引用了"tns:sayHello"元素。sayHelloResponse：描述了一个响应消息，结构与 sayHello 类似。

e) 端口类型定义:

My_HelloWebService：定义了 Web 服务的操作。operation：描述了名为"sayHello"的操作，包括输入和输出消息。

f) 绑定定义:

```
▼<binding name="My_HelloWebServicePortBinding" type="tns:My_HelloWebService">
   <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
  ▼<operation name="sayHello">
     <soap12:operation soapAction=""/>
    ▼<input>
       <soap12:body use="literal"/>
     </input>
    ▼<output>
       <soap12:body use="literal"/>
     </output>
   </operation>
 </binding>
```

My_HelloWebServicePortBinding 是针对 My_HelloWebService 端口类型的绑定，它指定了如何在 SOAP 1.2 协议上发送和接收这些消息。

g) 服务定义:

```
▼<service name="My_HelloWebService">
  ▼<port name="My_HelloWebServicePort" binding="tns:My_HelloWebServicePortBinding">
     <soap12:address location="http://localhost:7101/Demo/My_HelloWebServicePort"/>
   </port>
 </service>
```

"My_HelloWebService"服务提供了一个名为"My_HelloWebServicePort"的端口，该端口使用了前面定义的 My_HelloWebServicePortBinding 绑定，并且其地址位于 http://localhost:7101/Demo/My_HelloWebServicePort。

5) 思考问题：web service 有哪些类型？列出它们的主要异同点。

Web service 主要可以分为两大类：SOAP Web 服务和 RESTful Web 服务。

相同点：

- 分布式服务：无论是 SOAP 还是 REST，都是为分布式环境设计的，让不同的系统或应用可以通过网络互相通信。
- 基于 HTTP：在实际使用中，HTTP 是 SOAP 和 REST 最常用的传输协议。

- 独立于平台和语言：无论是 SOAP 还是 REST，它们都旨在提供与特定编程语言或操作系统无关的服务。例如，由 Java 编写的服务可以被 Python、.NET 或任何其他语言的客户端所调用。

不同点：

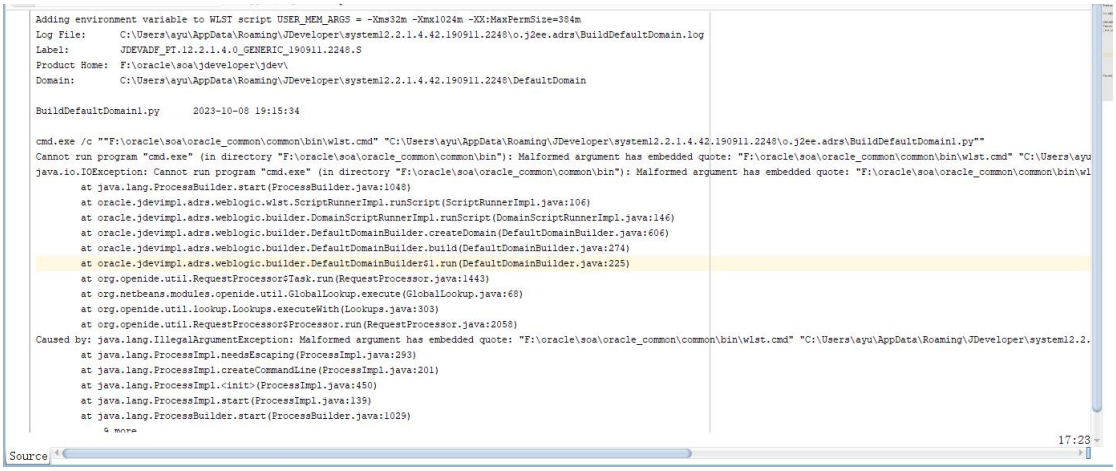
- 基于：SOAP 基于协议，而 REST 基于架构风格。
- 消息格式：SOAP 通常使用 XML，而 REST 可以使用多种格式。
- 绑定：SOAP 使用 WSDL 描述，而 REST 使用 URI 和 HTTP 动词。
- 错误处理：SOAP 使用特定的错误消息，而 REST 使用 HTTP 状态码。
- 性能和复杂性：SOAP 通常被视为更重、更复杂，而 REST 被视为更简单、更轻量级。

总而言之，SOAP 是一个重量级的服务交换协议，具有严格的规范和标准，适用于需要高级功能如事务和安全性的企业级应用。而 REST 是一个基于 HTTP 和标准 Web 技术的轻量级架构风格，适用于公开的 Web API 和移动应用等场景。

4.实验总结(完成的工作、对实验的认识、遇到的问题及解决方法)

在本次实验中，我实现了 JDeveloper 12c 中采用自底向上模式创建和测试 web service。通过本次实验，加深了我对 web service 的理解，为后续的学习奠定基础。

在本次实验中，启动默认域时遇到报错：



在查询网上资料后得到解决方法：

Follow the following steps to fix this issues in 12.2.1.4

1. Go to Oracle_Home\jdeveloper\ide\bin folder.
2. Edit ide.conf file.
3. Under ***# Other OSGi configuration options for locating bundles and boot delegation*** section (or any other section) add the following line

```
AddVMOption  -Djdk.lang.Process.allowAmbiguousCommands=true
```

4. Restart JDeveloper.

由此修改配置文件：

重启 JDeveloper 后成功启动默认域。