



面向服务的体系结构实验报告

实验名称：	实验二：创建 SOAP Web Services
实验日期：	2023-10-8
实验地点：	文宣楼 B312
提交日期：	2023-10-9

学号：	22920212204392
姓名：	黄勛
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

3.1 web service 有哪些类型？列出它们的主要异同点；

答：常见的Web服务类型：

1. SOAP (Simple Object Access Protocol):

- SOAP是一种基于XML的协议，用于在网络上交换结构化信息。
- 它支持多种传输协议，如HTTP、SMTP、TCP等。
- SOAP通常需要较大的消息头，因此它的消息相对较大，传输速度较慢。
- 通常与企业级应用和复杂系统集成中使用。

2. REST (Representational State Transfer):

- REST是一种基于HTTP的架构风格，它使用HTTP方法（如GET、POST、PUT、DELETE）来执行操作。
- REST使用轻量级的消息体，通常是JSON或XML格式。
- 它通常被认为更简单、更直观，适用于移动应用和Web应用的开发。
- REST是无状态的，每个请求都包含足够的信息以便服务器理解请求，不需要维护会话状态。

3. GraphQL:

- GraphQL是一种由Facebook开发的查询语言和运行时，用于获取精确的数据。
- 客户端可以定义其需要的数据结构，服务器将根据请求返回相应的数据。
- GraphQL减少了"过度获取"或"不足获取"的问题，提高了效率。
- 它适用于需要高度定制化数据请求的应用程序，如社交媒体平台和大型API。

4. gRPC:

- gRPC是Google开发的远程过程调用（RPC）框架，基于HTTP/2协议。
- 它使用Protocol Buffers（protobuf）来定义数据和服务接口。
- gRPC提供了高效的串行化和反串行化，支持多种编程语言。
- 适用于需要高性能通信和数据传输的场景，如微服务架构。

主要异同点：

- SOAP和REST之间的最大区别在于协议和消息格式。SOAP是一种协议，而REST是一种架构风格。SOAP使用XML消息格式，而REST通常使用JSON或XML。
- SOAP通常较复杂，适用于企业级应用，而REST更简单，适用于Web和移动应用。

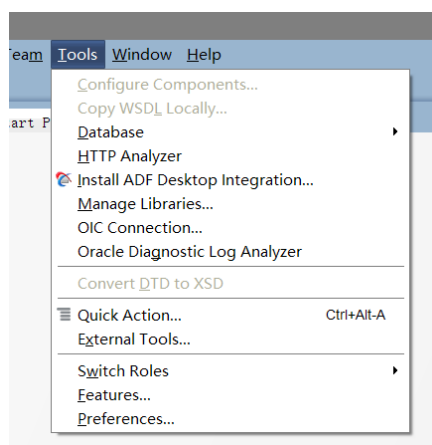
- GraphQL和REST都是基于HTTP的，但GraphQL允许客户端自定义请求，而REST使用固定的终端点。
- gRPC与SOAP和REST相比，更侧重于高性能和跨语言支持，因为它使用Protocol Buffers进行串行化。

不同的Web服务类型适用于不同的应用场景，选择取决于项目需求、性能要求。

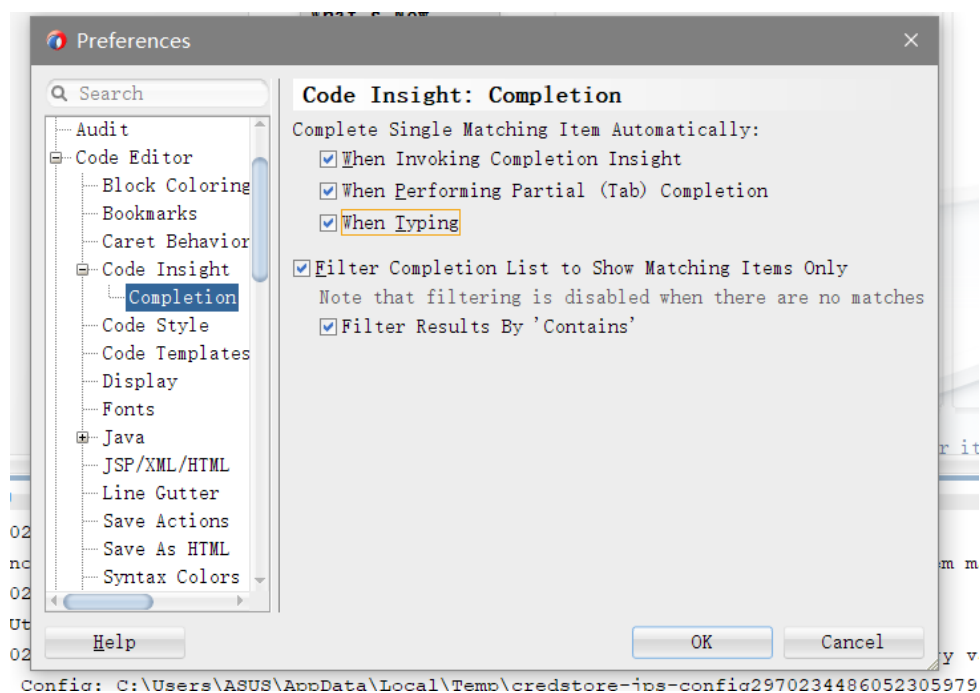
3.2 创建 SOAP web service 的步骤

3.2.1 JDeveloper 的代码补全功能设置（可选）

点击打开 JDeveloper > 点击菜单栏上的【Tools】



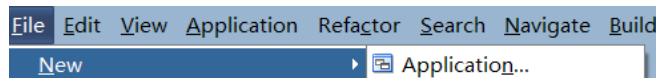
点击【Preferences...】，依次选择左侧栏的【code Editor】>【code insight】>【completion】>勾选右侧栏的【when typing】>点击【OK】关闭设置。



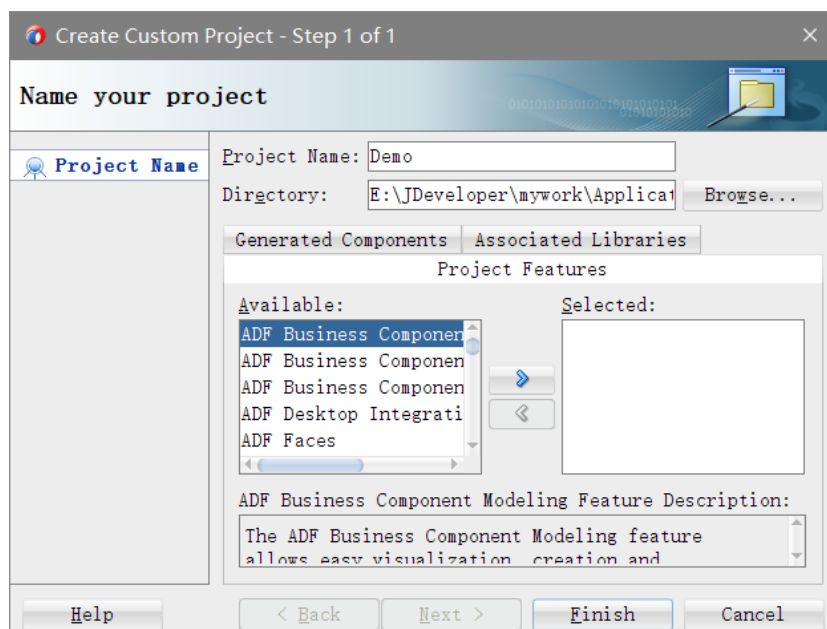
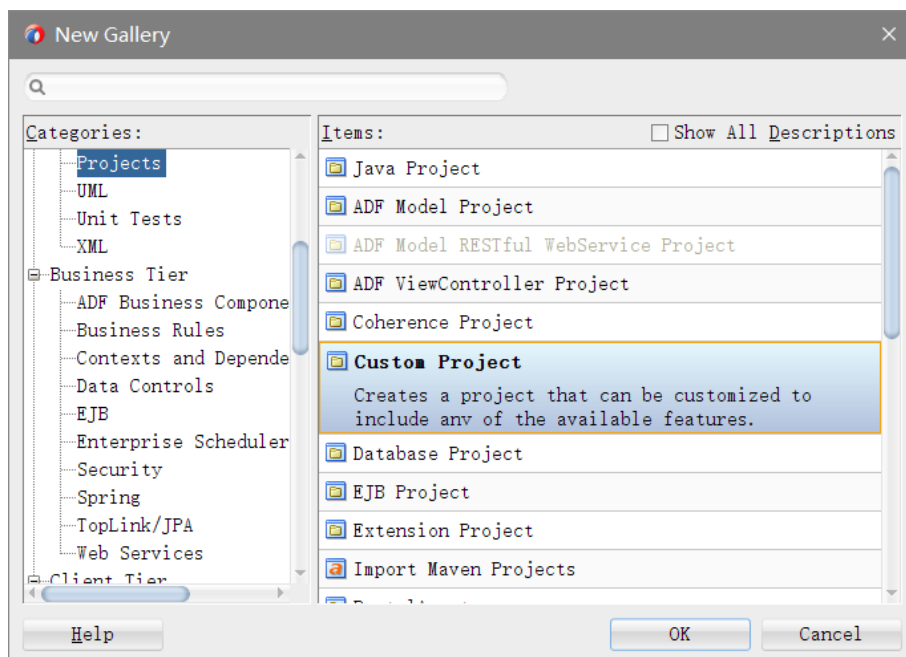
3.2.2 使用 JDeveloper 创建 web services

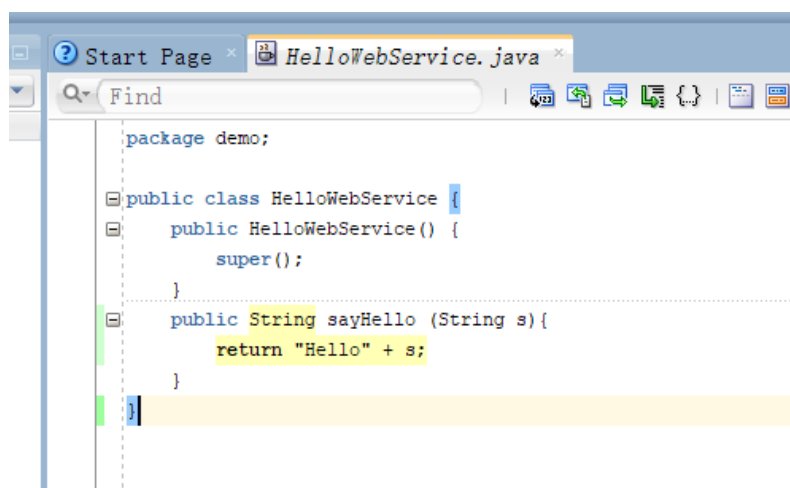
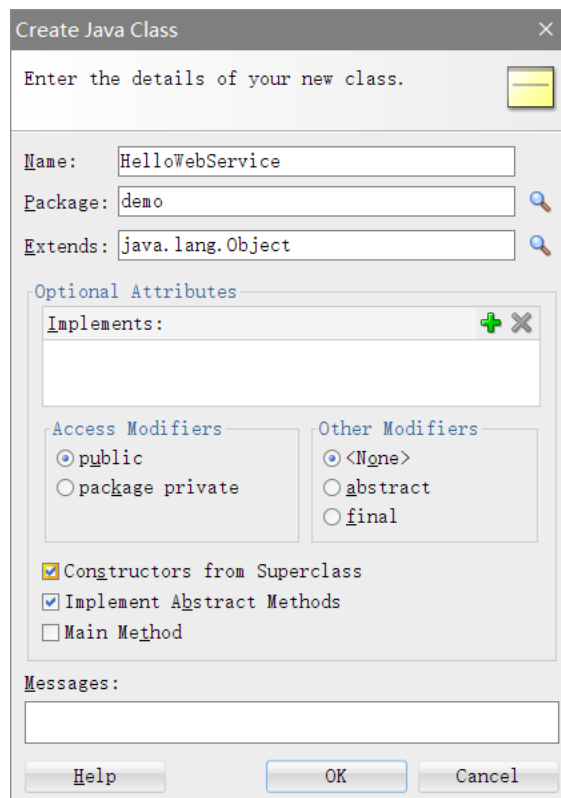
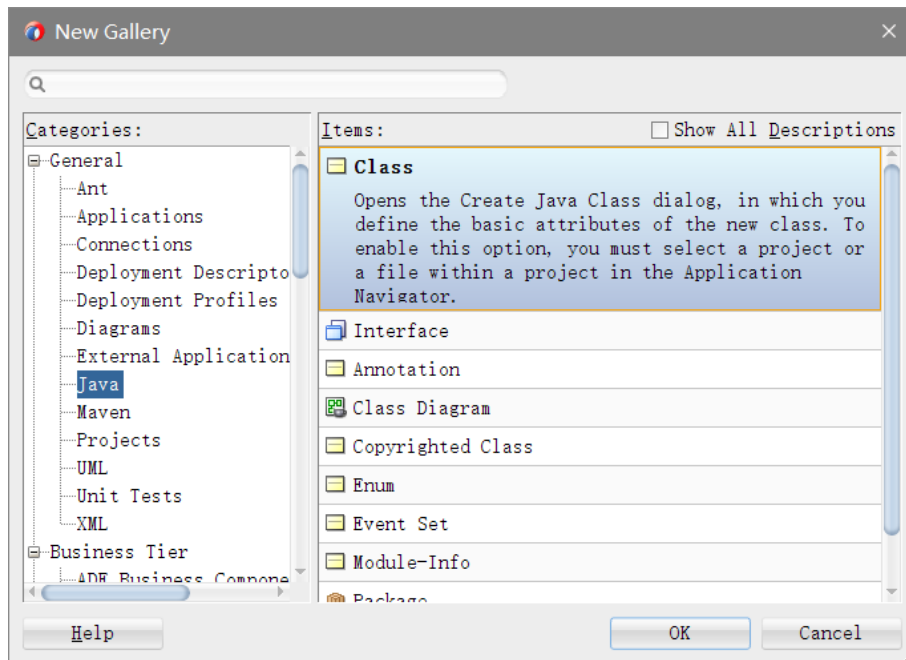
3.2.2.1 创建一个 Plain Old Java Object (POJO) 对象

创建一个 Application



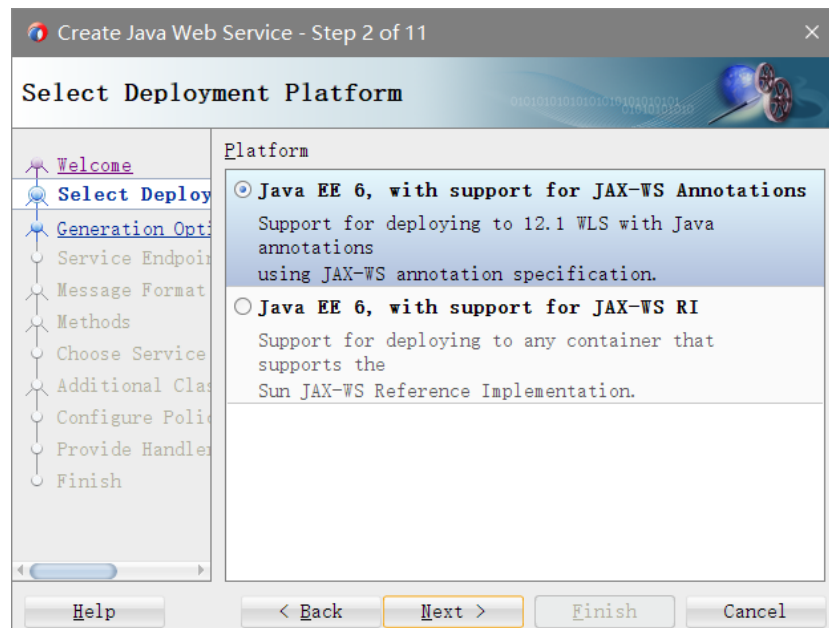
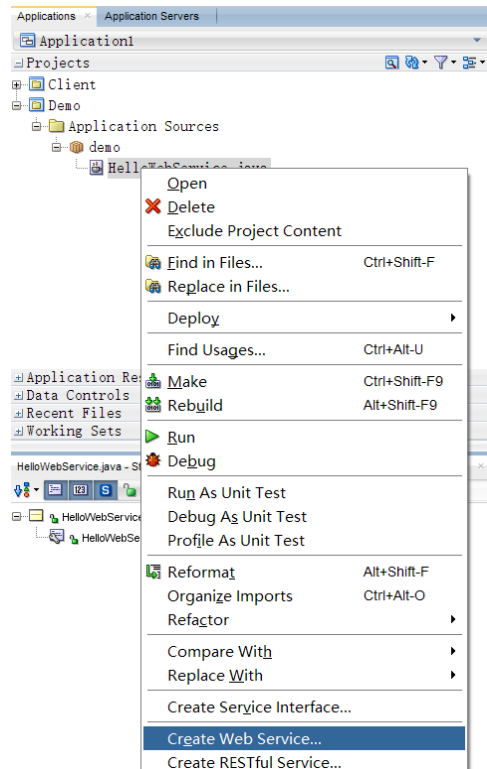
再在该 Application 下创建一个 Project (Custom Project)

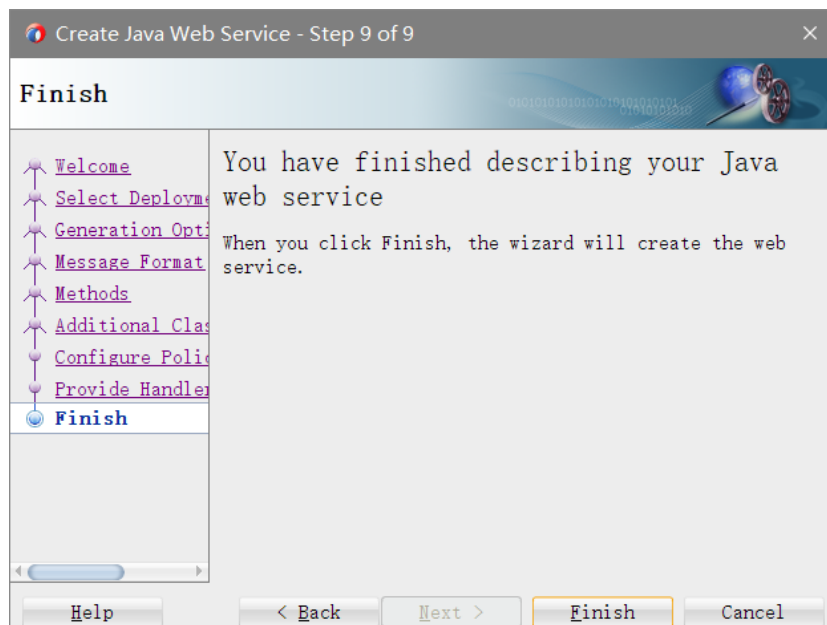
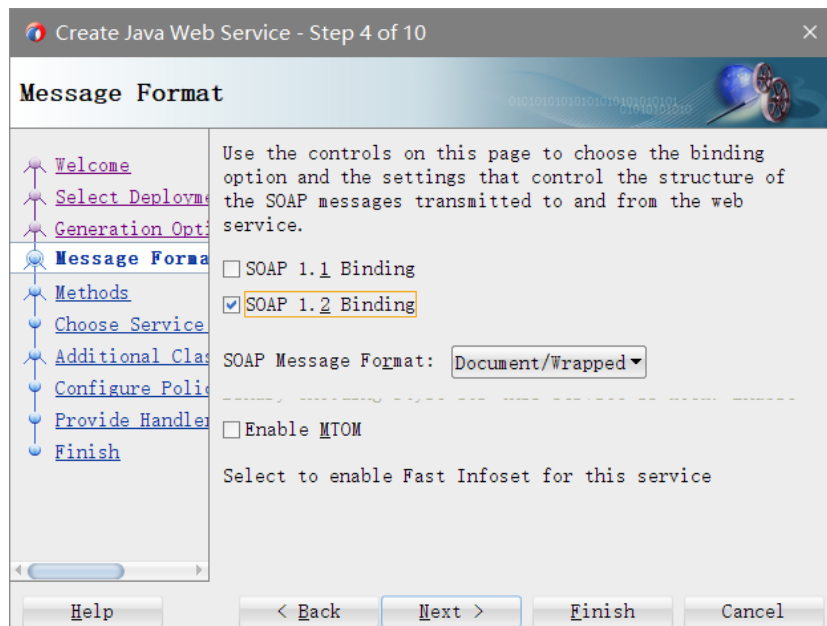
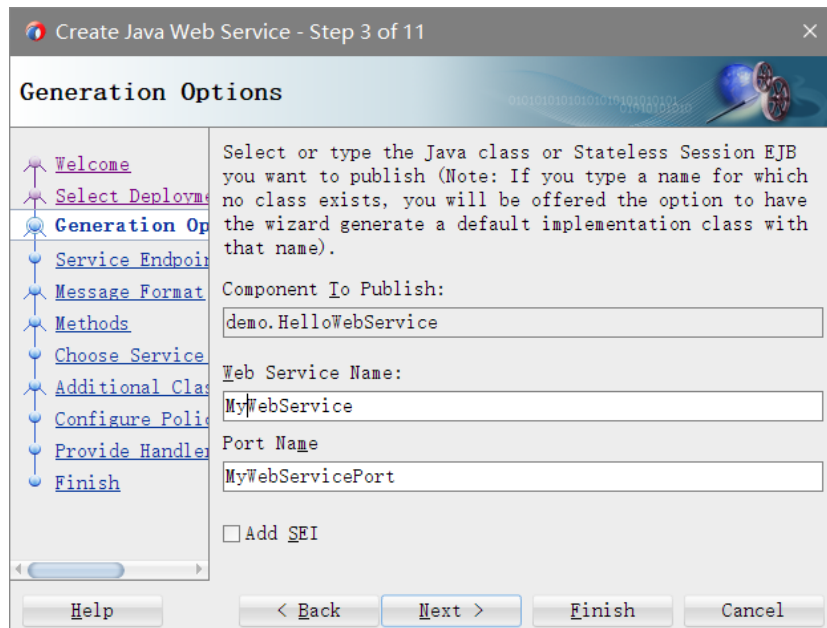




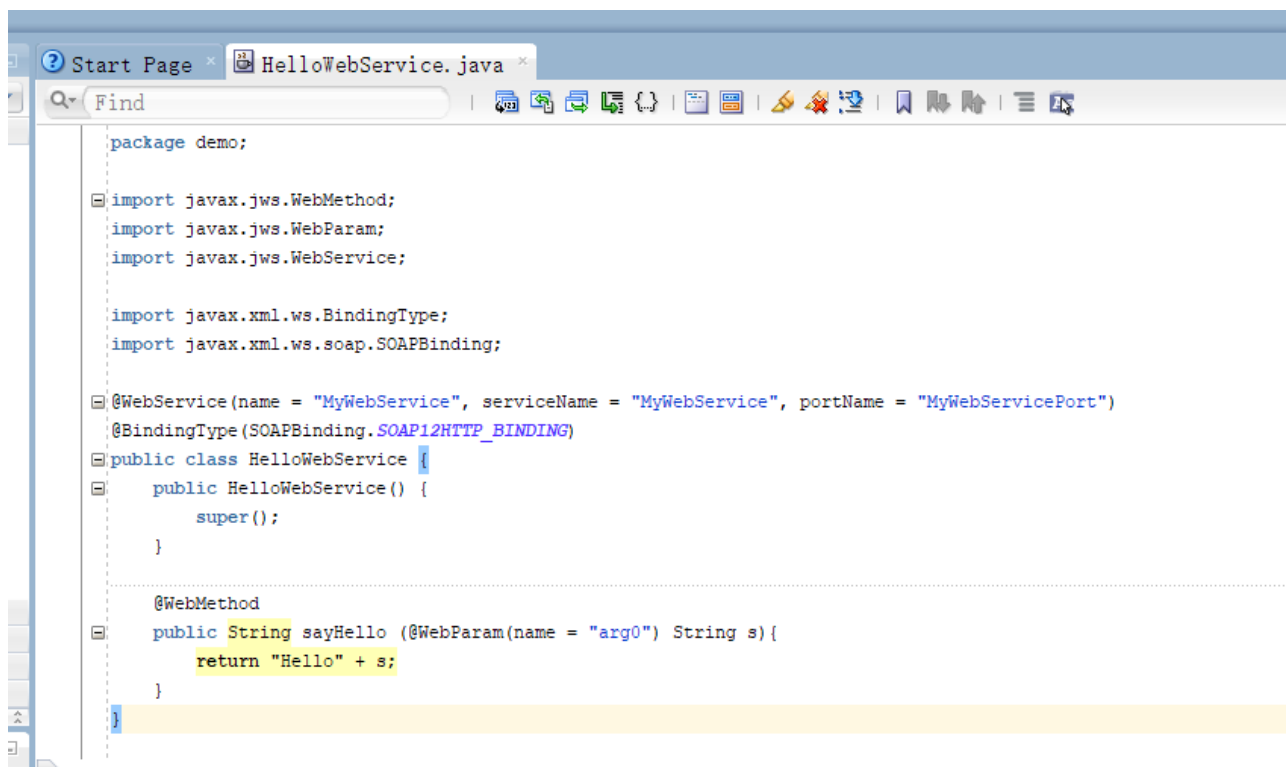
3.2.2.2 创建 web service 类

依次选择【Application Sources】>【demo】，右键【HellowebService.java】>【create web service...】。





完成后，原 java 类中添加了新的注释 annotations: @WebService、@BindingType、@WebMethod 和 @WebParam，保存。



```
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;

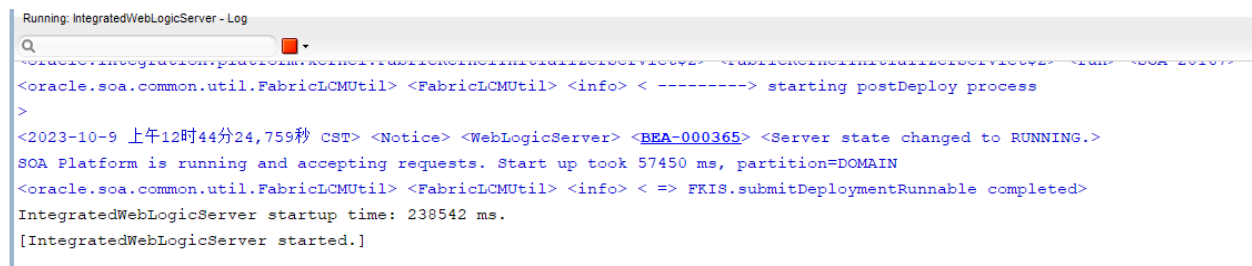
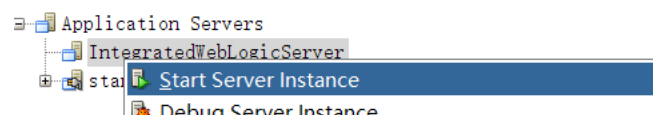
@WebService(name = "MyWebService", serviceName = "MyWebService", portName = "MyWebServicePort")
@BindingType(SOAPBinding.SOAP12HTTP_BINDING)
public class HelloWebService {
    public HelloWebService() {
        super();
    }

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

3.3 测试所创建的 web service 是否成功

3.3.1 启动 IntegratedWebLogicServer

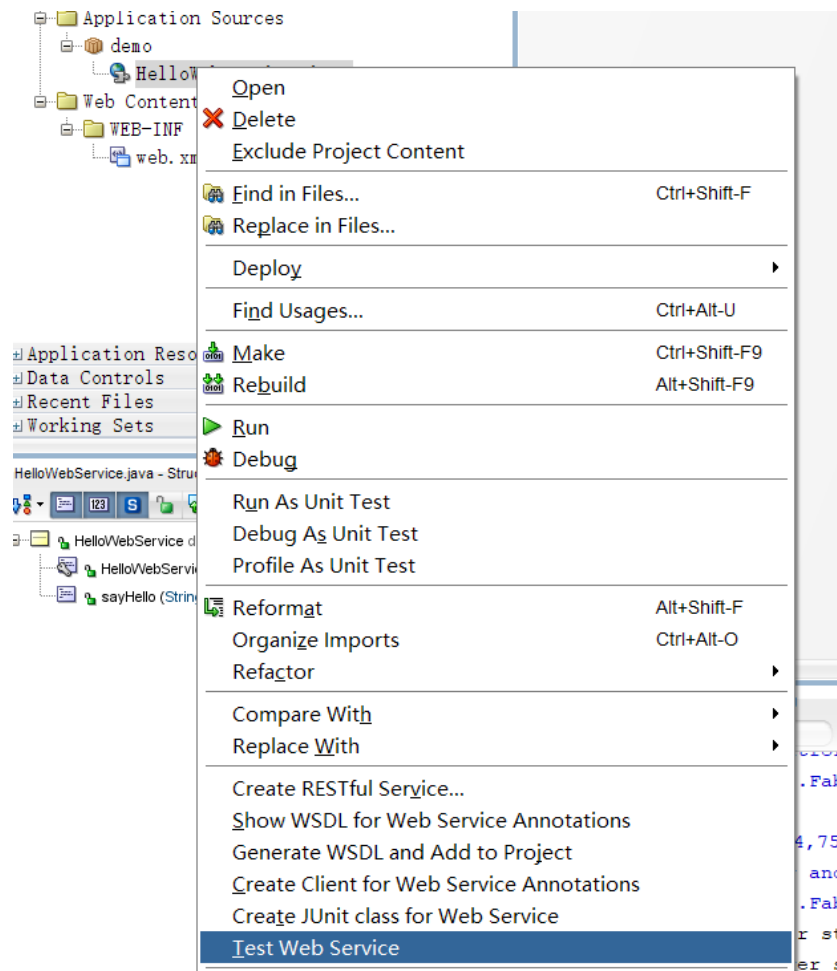
依次选择【Application servers】>【IntegratedWebLogicServer】>【Start server instance】。



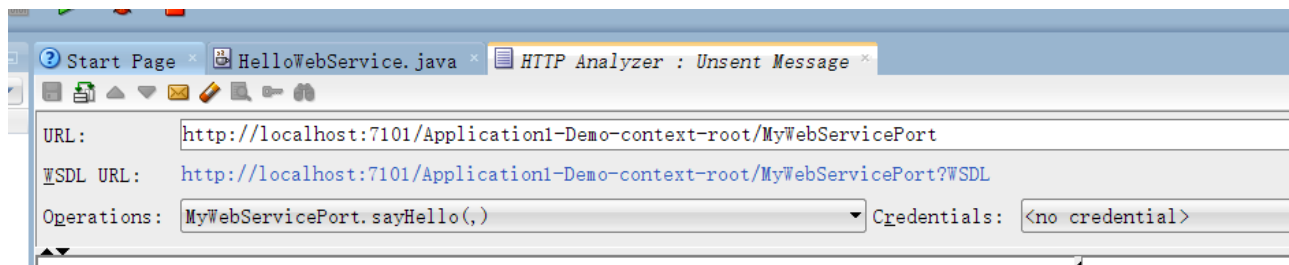
```
Running: IntegratedWebLogicServer - Log
<oracle.integration.platform.kernel.FabricKernelInitializerService> <FabricKernelInitializerService> <init> SOA-2010-
<oracle.soa.common.util.FabricLCMUtil> <FabricLCMUtil> <info> <-----> starting postDeploy process
>
<2023-10-9 上午12时44分24,759秒 CST> <Notice> <WebLogicServer> <BEA-000365> <Server state changed to RUNNING.>
SOA Platform is running and accepting requests. Start up took 57450 ms, partition=DOMAIN
<oracle.soa.common.util.FabricLCMUtil> <FabricLCMUtil> <info> <=> FKIS.submitDeploymentRunnable completed>
IntegratedWebLogicServer startup time: 238542 ms.
[IntegratedWebLogicServer started.]
```

3.3.2 测试 web service

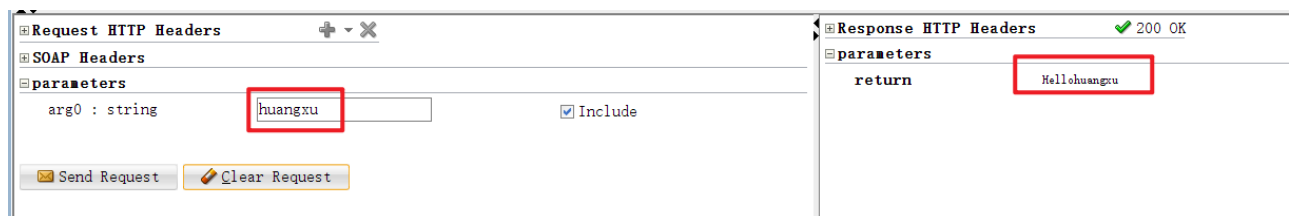
在应用窗口右键【HelloWebService.java】>【Test Web Service】。



在 HTTP Analyzer 编辑器中显示了 web service 的 URL，WSDL URL 和暴露的操作。



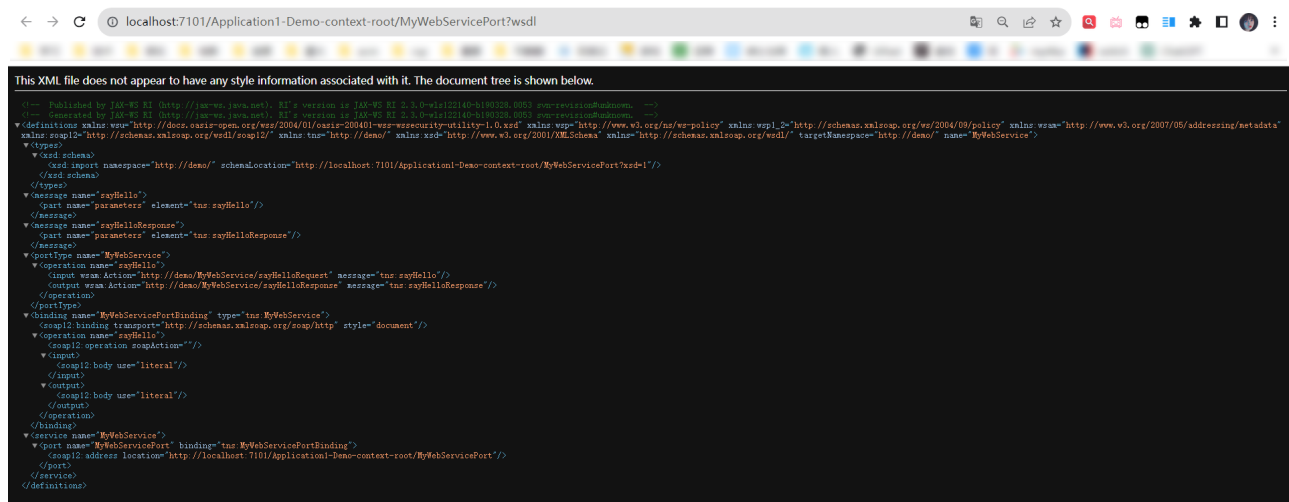
在请求区域，在 arg0 字段输入名字 `huangxu` 点击【send request】，在响应区域就可以看到结果。



至此，自底向上创建 web service 的方法就结束了。

3.4 在文本编辑器或浏览器中打开 WSDL 文档，分析 WSDL 的各部分内容

打开链接 <http://localhost:7101/Application1-Demo-context-root/MyWebServicePort?wsdl>



1. 命名空间定义：

- 这个文档使用了多个命名空间，以不同的前缀来标识不同的XML元素。例如，`xmlns:wsu`、`xmlns:wsp`、`xmlns:wsp1_2`等。
- `targetNamespace="http://demo/"` 定义了文档的目标命名空间为 "<http://demo/>"，这是这个Web服务的命名空间。

2. `<types>` 元素：

- 这个部分定义了XML模式（XML Schema），用于描述Web服务的消息结构。
- 在这里，使用 `<xsd:schema>` 引入了一个XML模式（Schema），该模式定义了位于 "<http://demo/>" 命名空间下的消息结构。

3. `<message>` 元素：

- 定义了两个消息，分别是 "sayHello" 和 "sayHelloResponse"。
- 这些消息指定了消息元素的名称和命名空间，以及它们在操作中的角色。

4. `<portType>` 元素：

- 定义了一个名为 "MyWebService" 的端口类型，其中包含一个名为 "sayHello" 的操作。
- 操作 "sayHello" 具有输入和输出消息，分别指定了操作的输入和输出参数。

5. `<binding>` 元素：

- 定义了一个名为 "MyWebServicePortBinding" 的绑定类型，它绑定到上述定义的端口类型 "MyWebService"。
- 使用 SOAP 1.2 协议进行绑定，指定了消息的传输协议和消息样式。

6. `<service>` 元素：

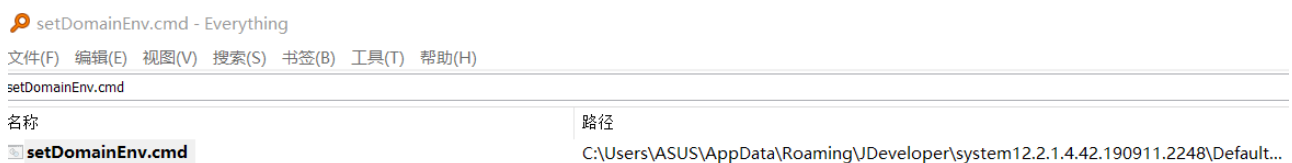
- 定义了一个名为 "MyWebService" 的服务，该服务包含一个端口 "MyWebServicePort"。
- 该端口使用了之前定义的绑定类型 "MyWebServicePortBinding"，并指定了服务的访问地址。

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)

本次实验初步了解了如何使用创建 java 类自底而上地实现 SOAP web service，也初步了解了 Web 服务的相关基础知识。

4.1 出现The JRE was not found in directory D:\Program Files\Java\jdk1.8.0_301.

需要修改domain中的 setDomainEnv.cmd 文档，将SUN_JAVA_HOME 和 JAVA_HOME 修改为安装后的JDK路径。



```
set SUN_JAVA_HOME=D:\Java\jdk1.8.0_301
```

```
set DEFAULT_SUN_JAVA_HOME=D:\Java\jdk1.8.0_301
```

```
set JAVA_HOME=D:\Java\jdk1.8.0_301
```

4.2 理解并分析 HTTP Analyzer 编辑器中显示的 web service 的 URL，WSDL URL 和暴露的操作。

URL：Web 服务的 URL 是指可以用于访问该服务的统一资源定位符。在 HTTP Analyzer 编辑器中，可以查看 Web 服务的 URL，该 URL 可以用于发送请求和接收响应。通过分析 Web 服务的 URL，可以确定服务的位置和访问方式。

WSDL URL：WSDL (Web Services Description Language) 是一种用于描述 Web 服务接口和消息格式的 XML 文件。WSDL URL 是指用于访问 Web 服务的描述文件 (WSDL 文件) 的 URL。在 HTTP Analyzer 编辑器中，可以查看 Web 服务的 WSDL URL，通过访问该 URL，可以获取有关服务接口、操作和数据类型的详细信息。

暴露的操作：在 HTTP Analyzer 编辑器中，可以查看 Web 服务所暴露的操作。操作是指可以通过 Web 服务执行的特定功能或任务。通过分析暴露的操作，可以了解可以使用该服务执行的不同操作，并确定它们的输入参数和输出结果。