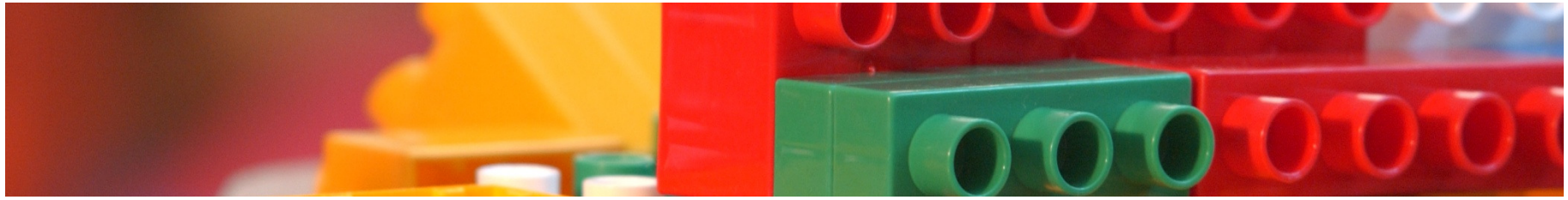
Introduction to Oracle WebLogic

Presented by: Fatna Belqasmi,
PhD, Researcher at Ericsson



Agenda

- Overview
- Download and installation
- A concrete scenario using the real product
- Hints for the project

Overview

- Oracle WebLogic Web Server
- Oracle Workshop for WebLogic

Oracle WebLogic Web Server

- Is a Java Enterprise Edition (Java EE) application server
- Supports the deployment of many types of distributed applications
- Provides a standard set of APIs for creating distributed Java applications that can access a wide variety of services
  - Web Services
  - Web applications (HTML, Java ServerPages-JSP, ..)
  - Remote Method Invocation (RMI)
  - Java Database Connectivity (JDBC)
  - ….

Oracle WebLogic Web Server

- WebLogic Server 10.3 supports two types of Web Service APIs
  - Java API for XML based Remote Procedure Call (JAX-RPC)
    - Defines the Java APIs for making XML-based remote procedure calls (RPC)
  - Java API for XML based Web Services (JAX-WS)
    - Is a standards-based API for coding, assembling, and deploying Java Web Services
    - Is the successor to the JAX-RPC

Oracle WebLogic Web Server

- WebLogic Server 10.3 supports the following Web Service standards
  - JAX-WS 2.1
  - JAX-RPC 1.1
  - Web Services for Java EE 1.2
  - Web Services Metadata for the Java Platform 2.0 (JSR-181)
  - Web Services Description Language (WSDL) 1.1
  - Simple Object Access Protocol (SOAP) 1.1 and 1.2
  - Web Services Security (WS-Security) 1.1
  - Universal Description, Discovery, and Integration (UDDI) 2.0
  - …

Oracle Workshop for WebLogic

- Is a set of plug-ins to the Eclipse Integrated Development Environment (IDE) platform
  - Many of the standard features of the workshop are described in the Eclipse documentation, available at http://eclipse.org
- Allows quick and easy creation, deployment and testing of enterprise applications:
  - Web Services
  - Java
  - Web Applications
  - …
- The applications can be deployed on a wide variety of servers, including
  - WebLogic Server
  - Tomcat,
  - JBoss, and others

Download and installation

- Download
  - Free download with registration

http://www.oracle.com/technology/software/products/ias/bea_main.html

  - Accept the license agreement on the top of the page
  - Click the link:
    - "Oracle Workshop for WebLogic 10.3 - Package Installer" OR
    - "Oracle Workshop for WebLogic 10.3 - Net Installer"
  - Sign in or register if you are new

- Installation
  - Double click the downloaded file and follow the instructions

- Documentations:
  - http://e-docs.bea.com/wlw/docs103/index.html
  - http://e-docs.bea.com/wls/docs103/webservices.html

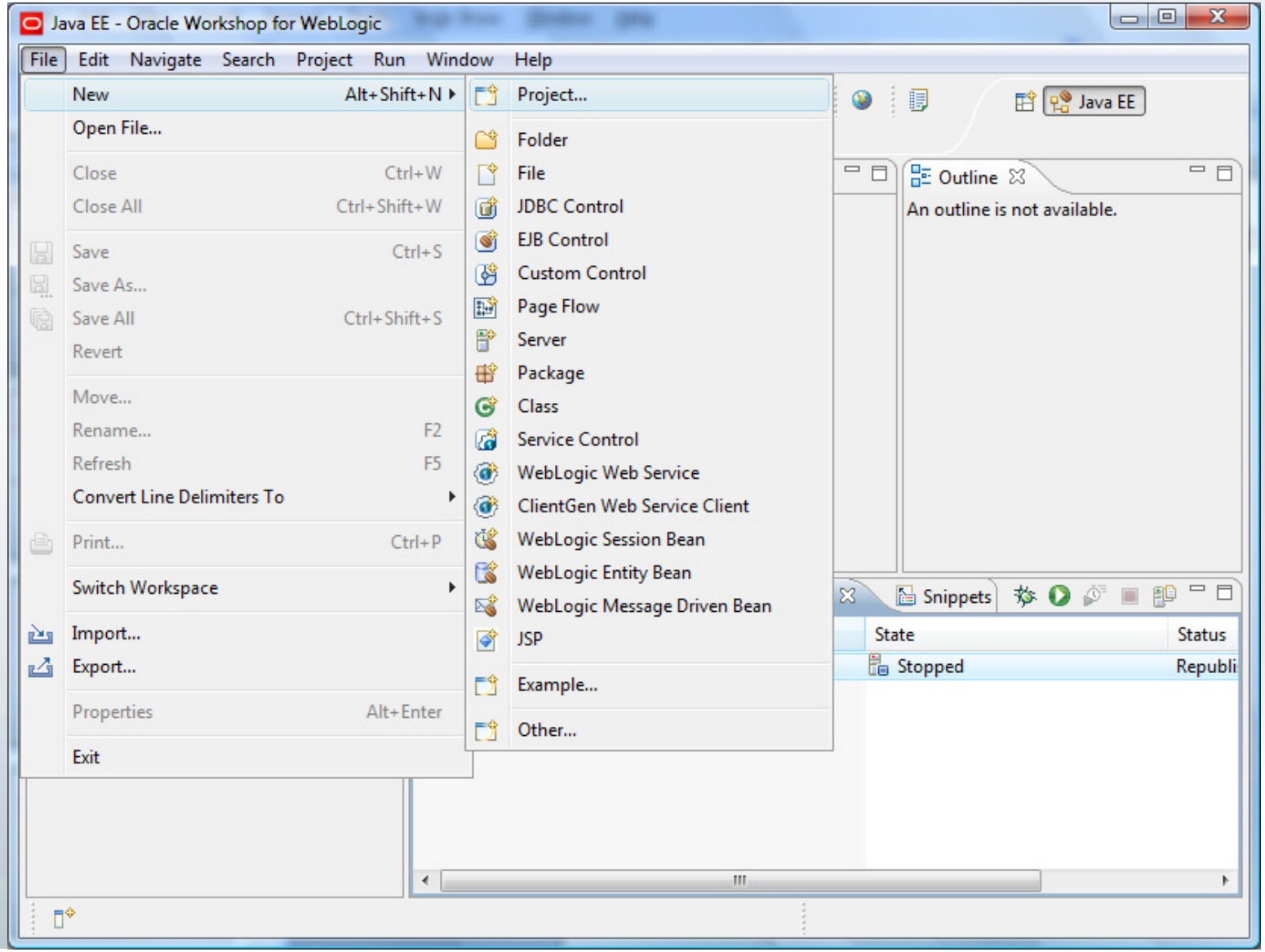
A Concrete Scenario Using the Real Product

A Concrete Scenario

- A simple “Hello World” web service
  - Create a Web Service
    - Create project
    - Create package
    - Create a Web Service
    - Add a web service method
  - Add/start weblogic server
    - Create a domain
    - Start server
  - Deploy and test the Web Service
    - Run on server
    - Web logic test client
      - See SOAP, WSDL

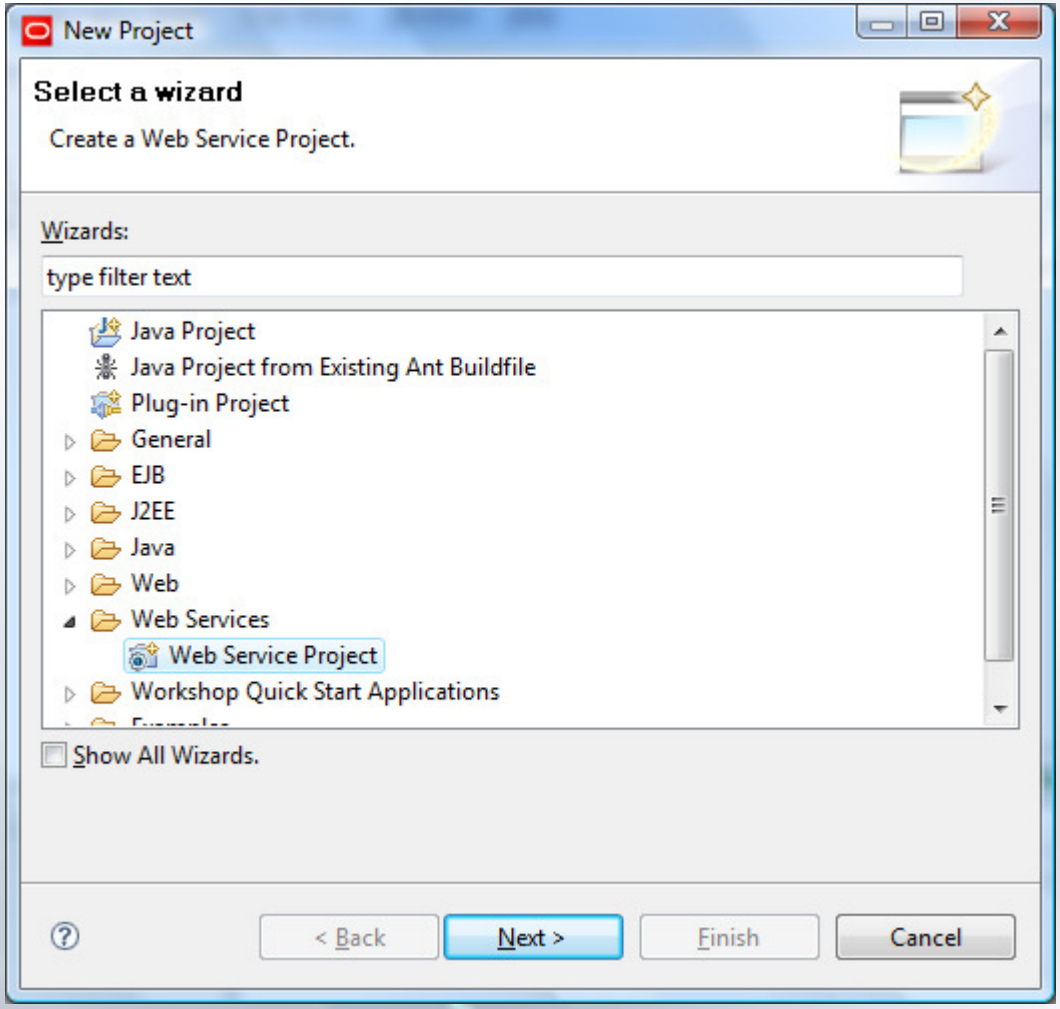
A Concrete Scenario

- Create a project



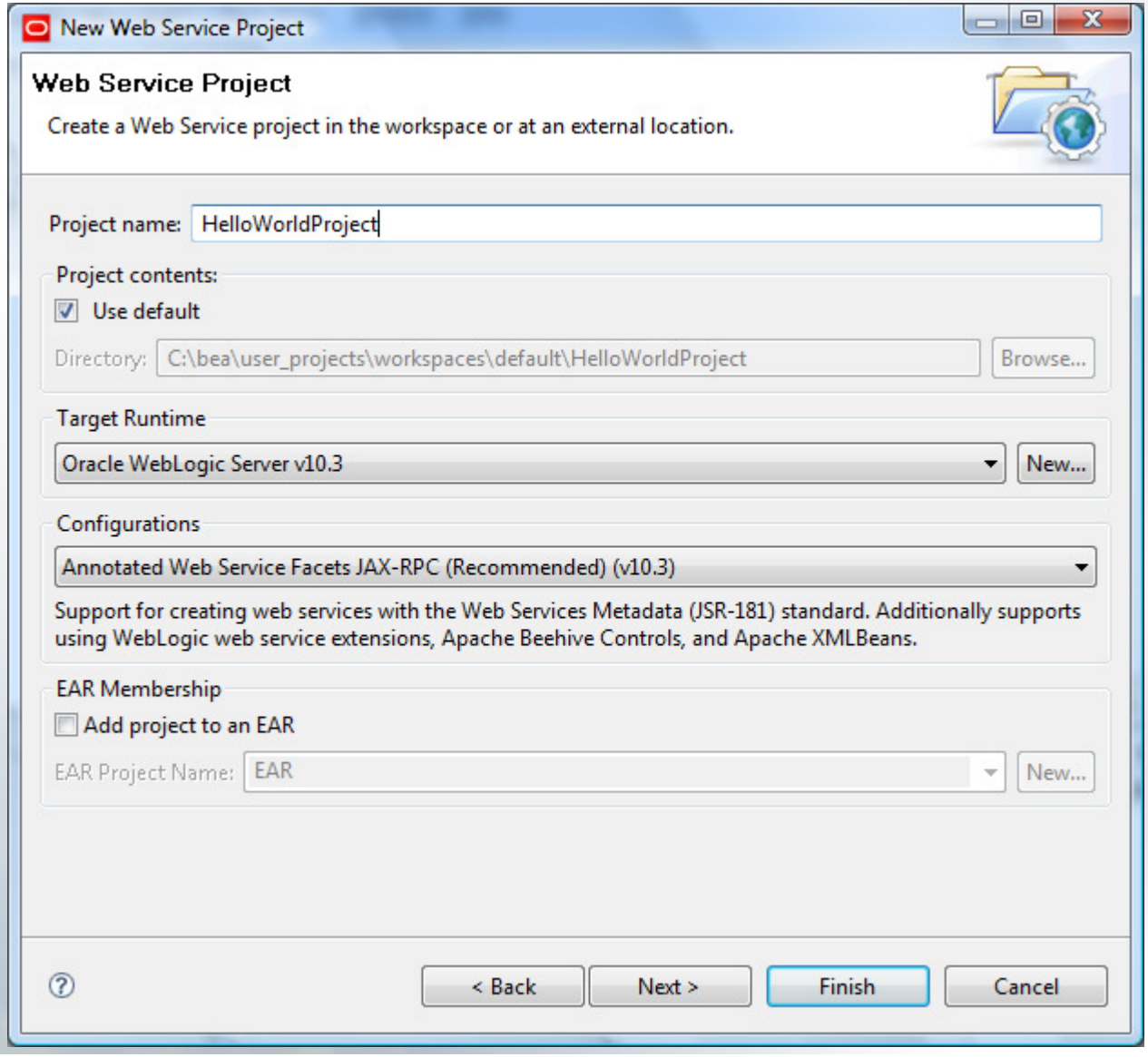
A Concrete Scenario

- Create a project



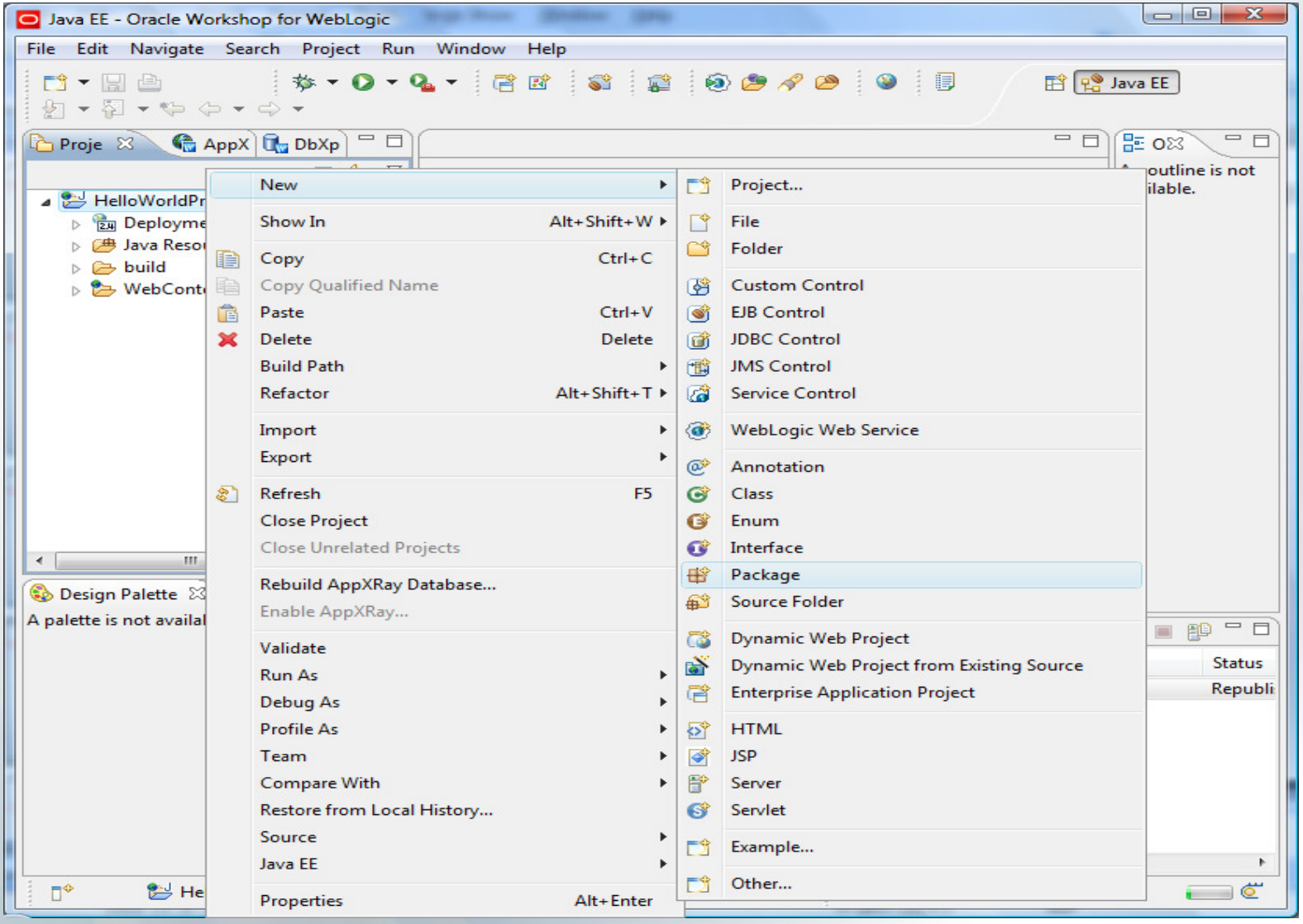
A Concrete Scenario

- Create a project



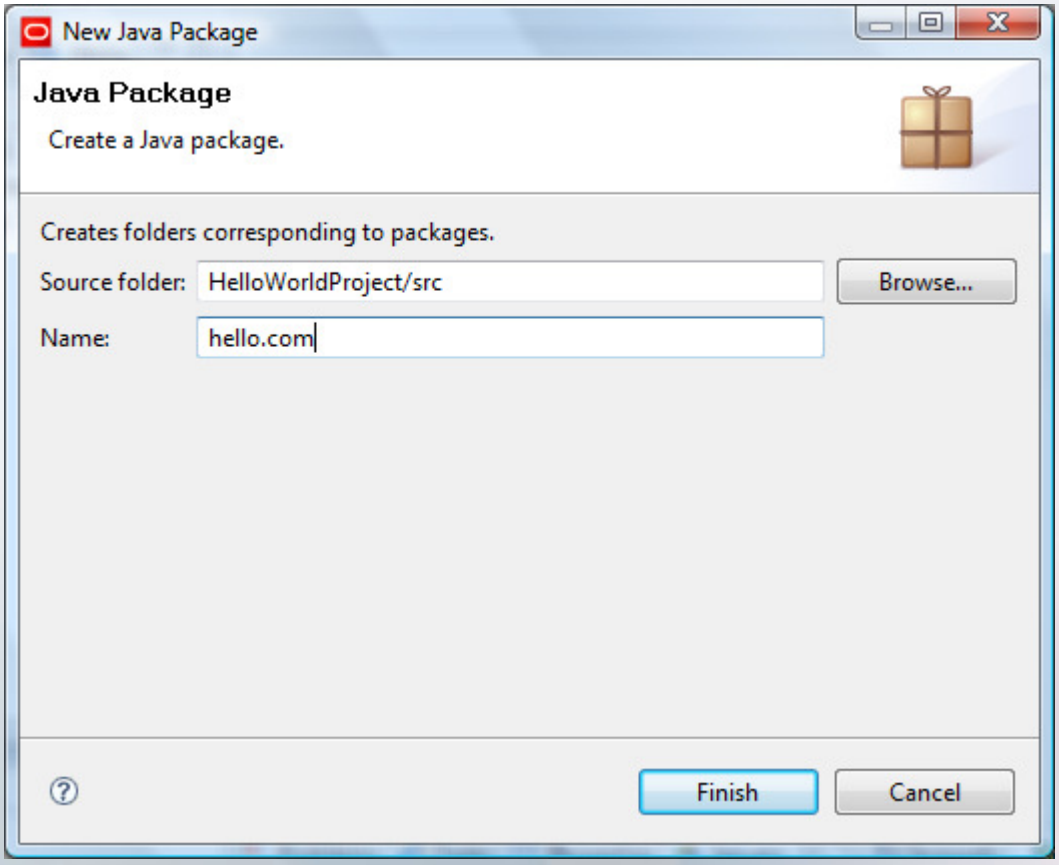
A Concrete Scenario

- Create a new package



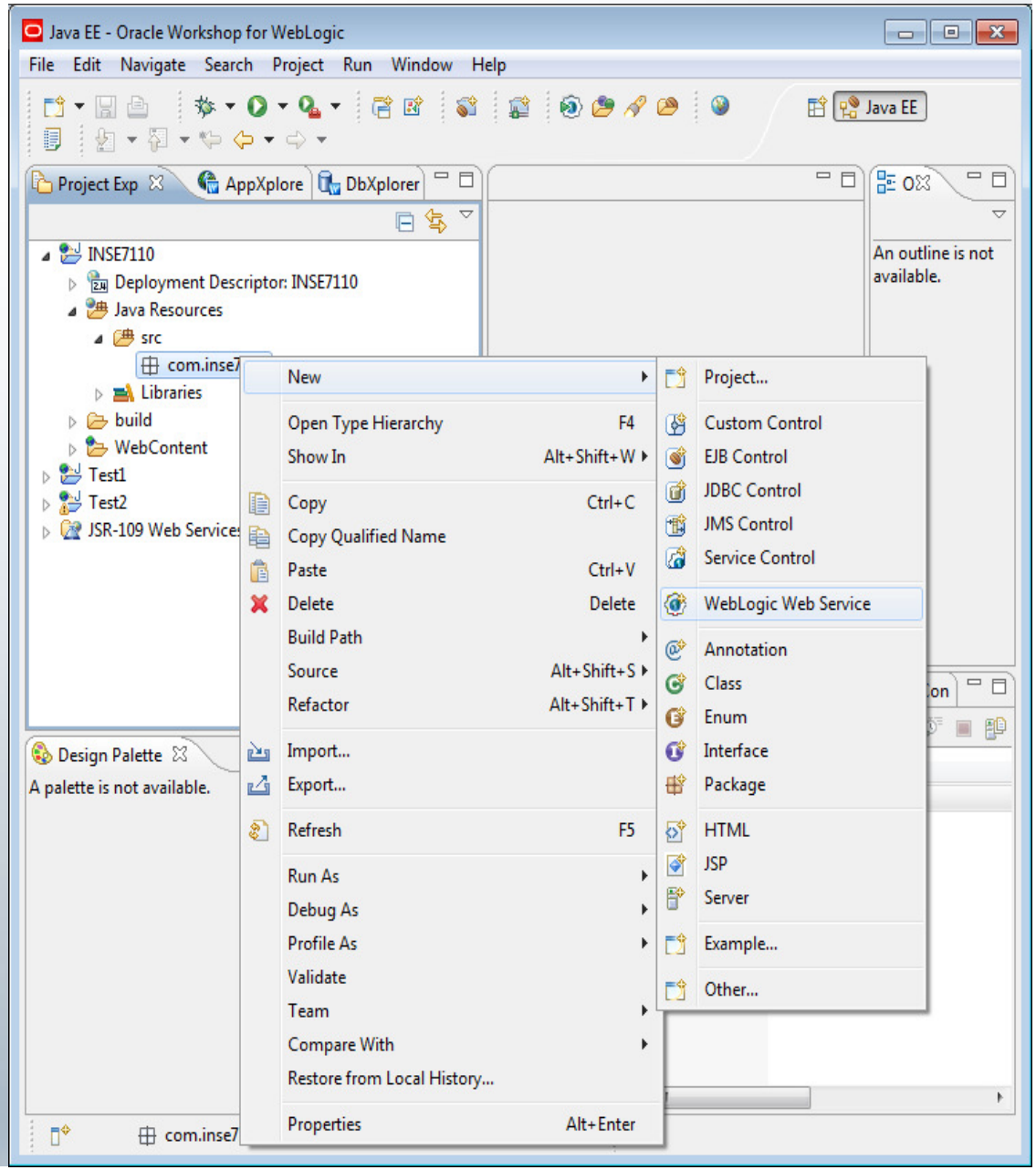
A Concrete Scenario

- Create a new package



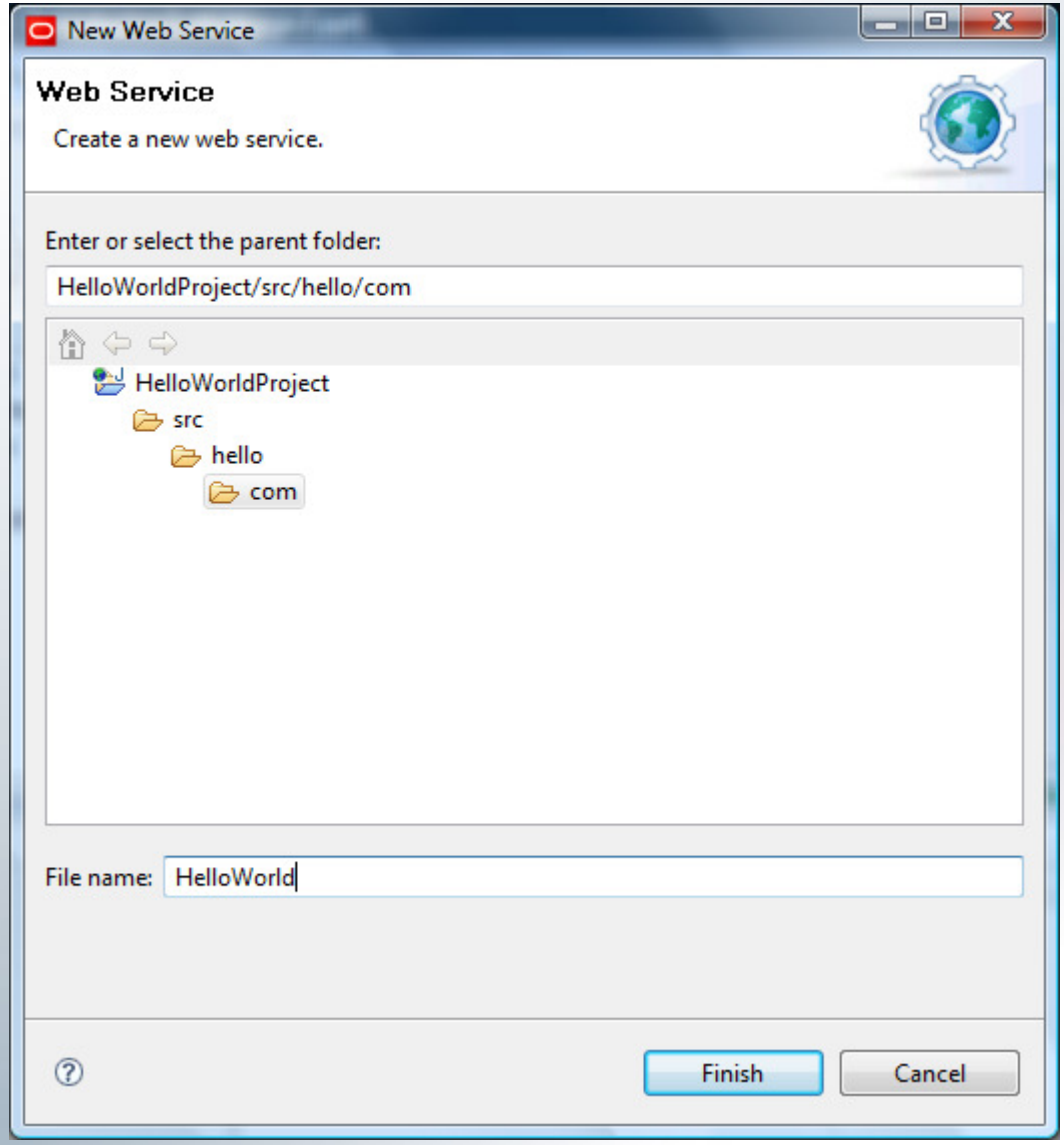
A Concrete Scenario

- Create a Web Service
  - Specify the names and parameters of all of the service's exposed operations
- Steps
  - Create a WebLogic WS
  - Add the methods
  - Configure the methods' parameters



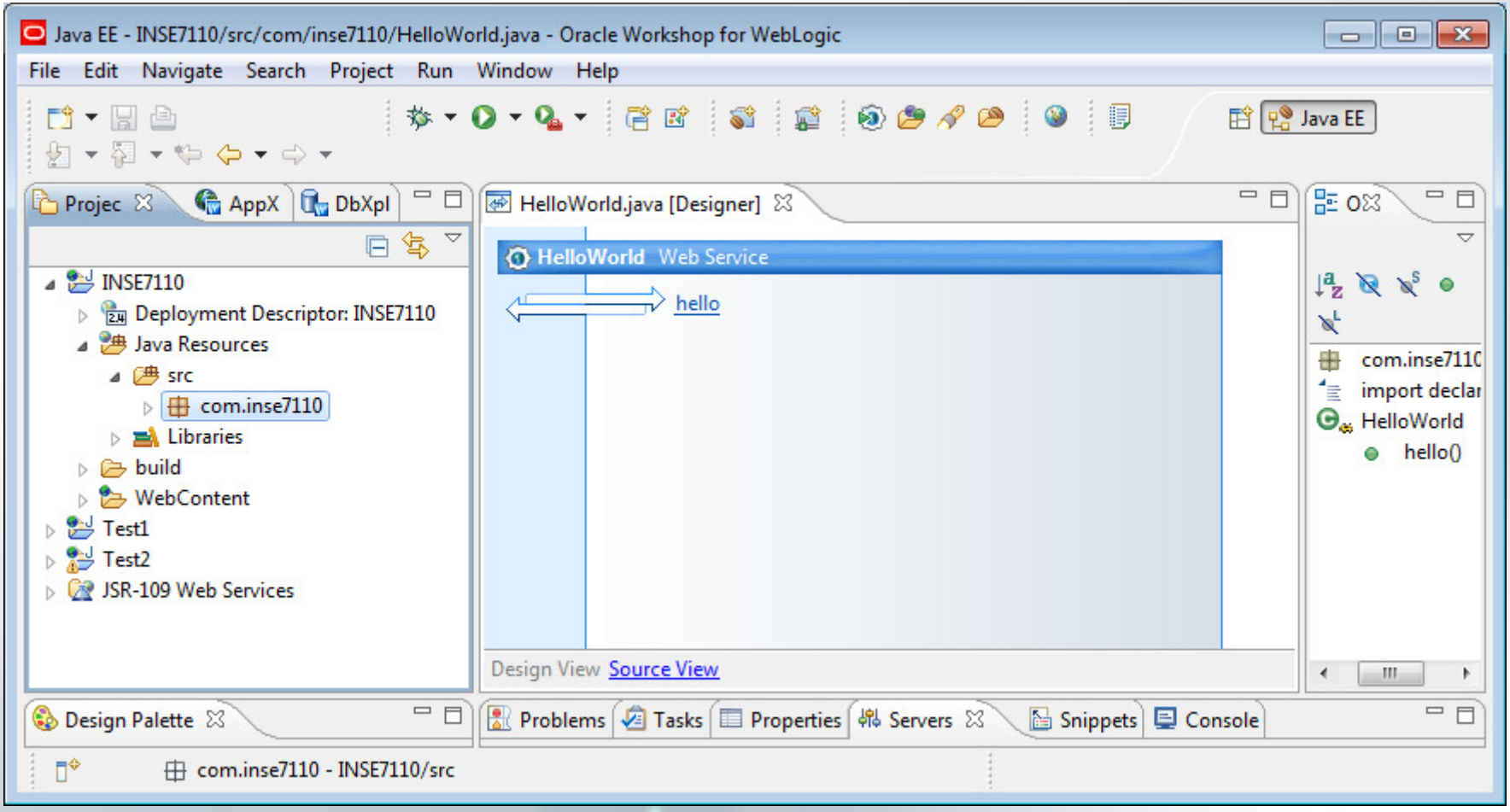
A Concrete Scenario

- Create a Web Service



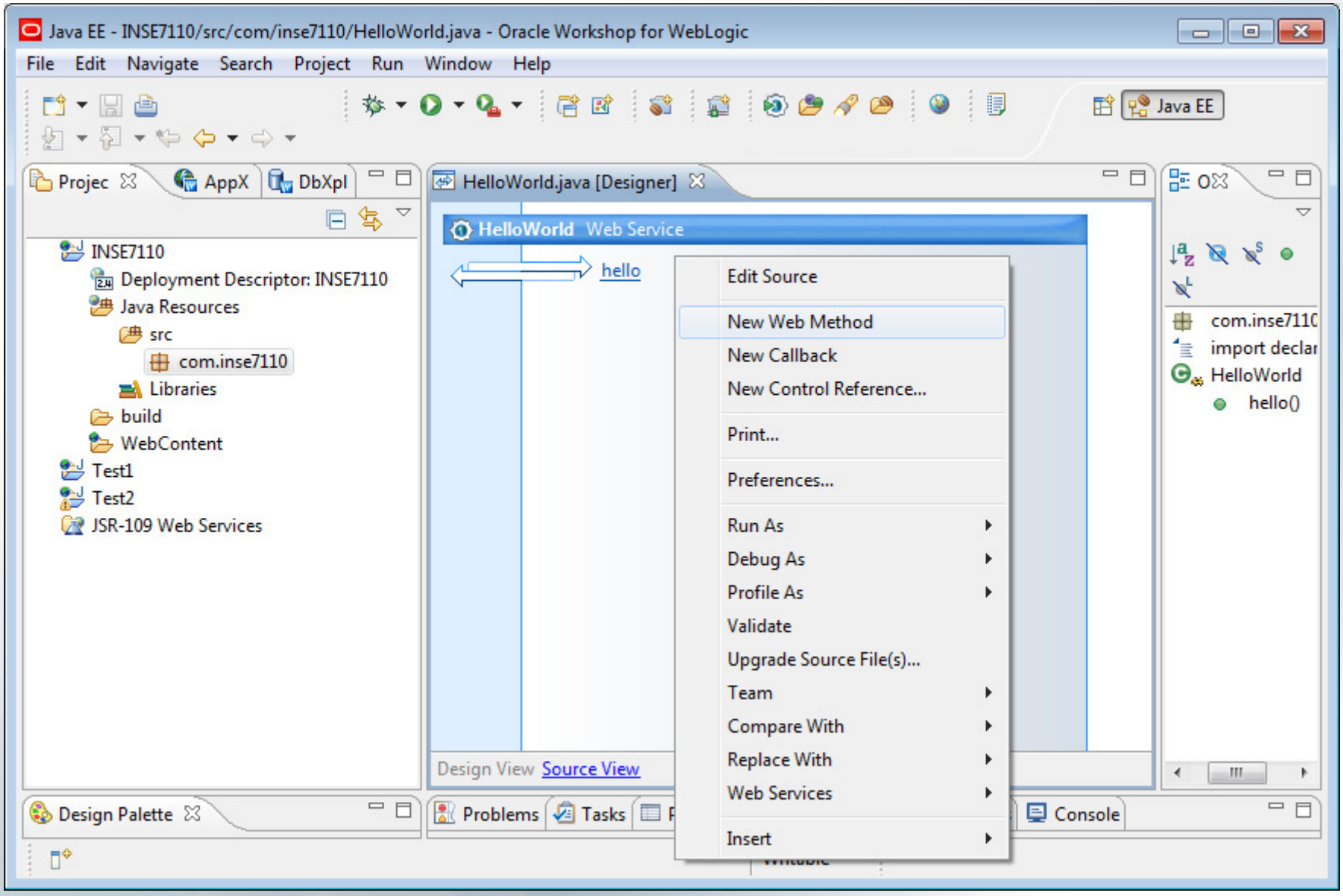
A Concrete Scenario

- Create a Web Service



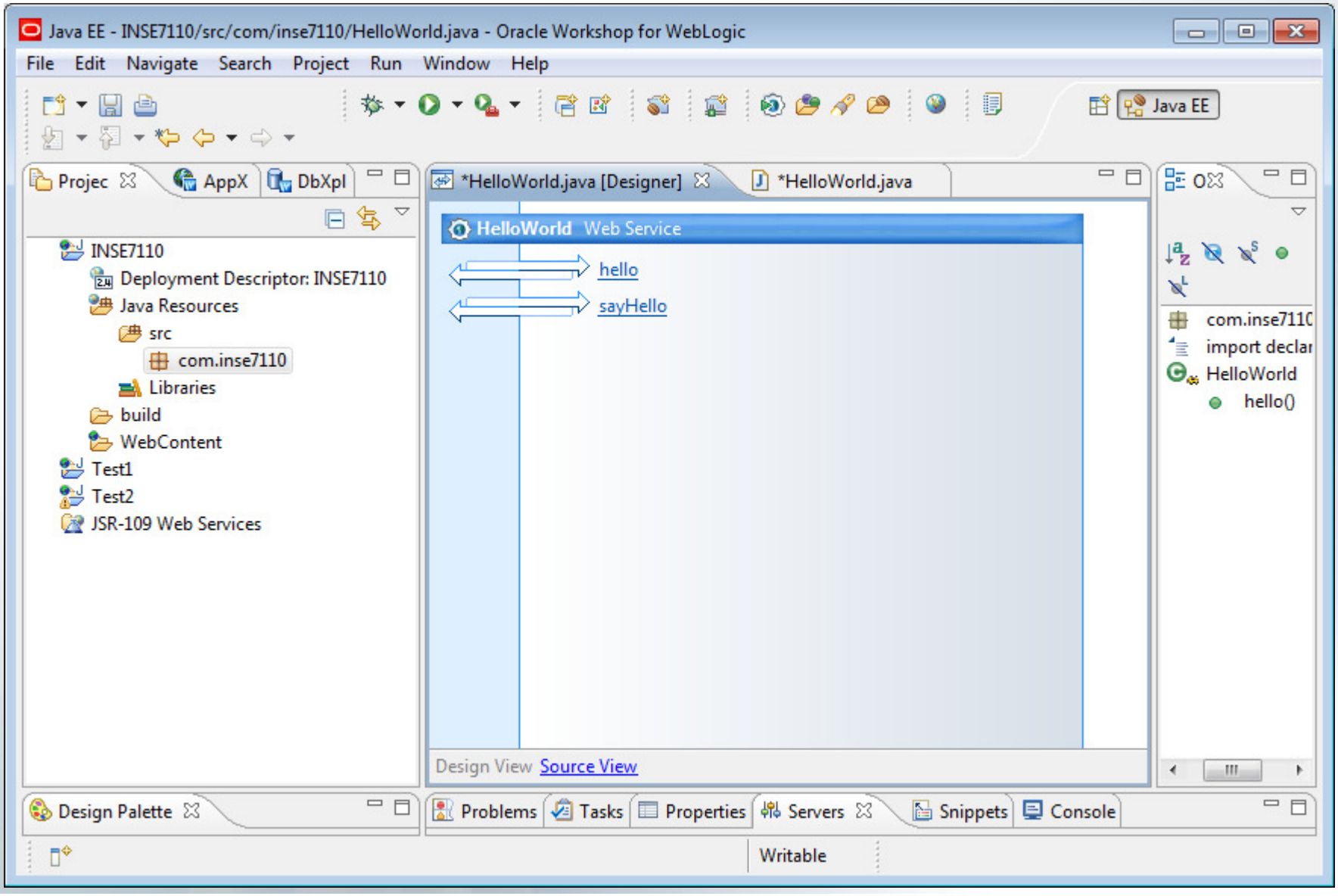
A Concrete Scenario

- Create a Web Service



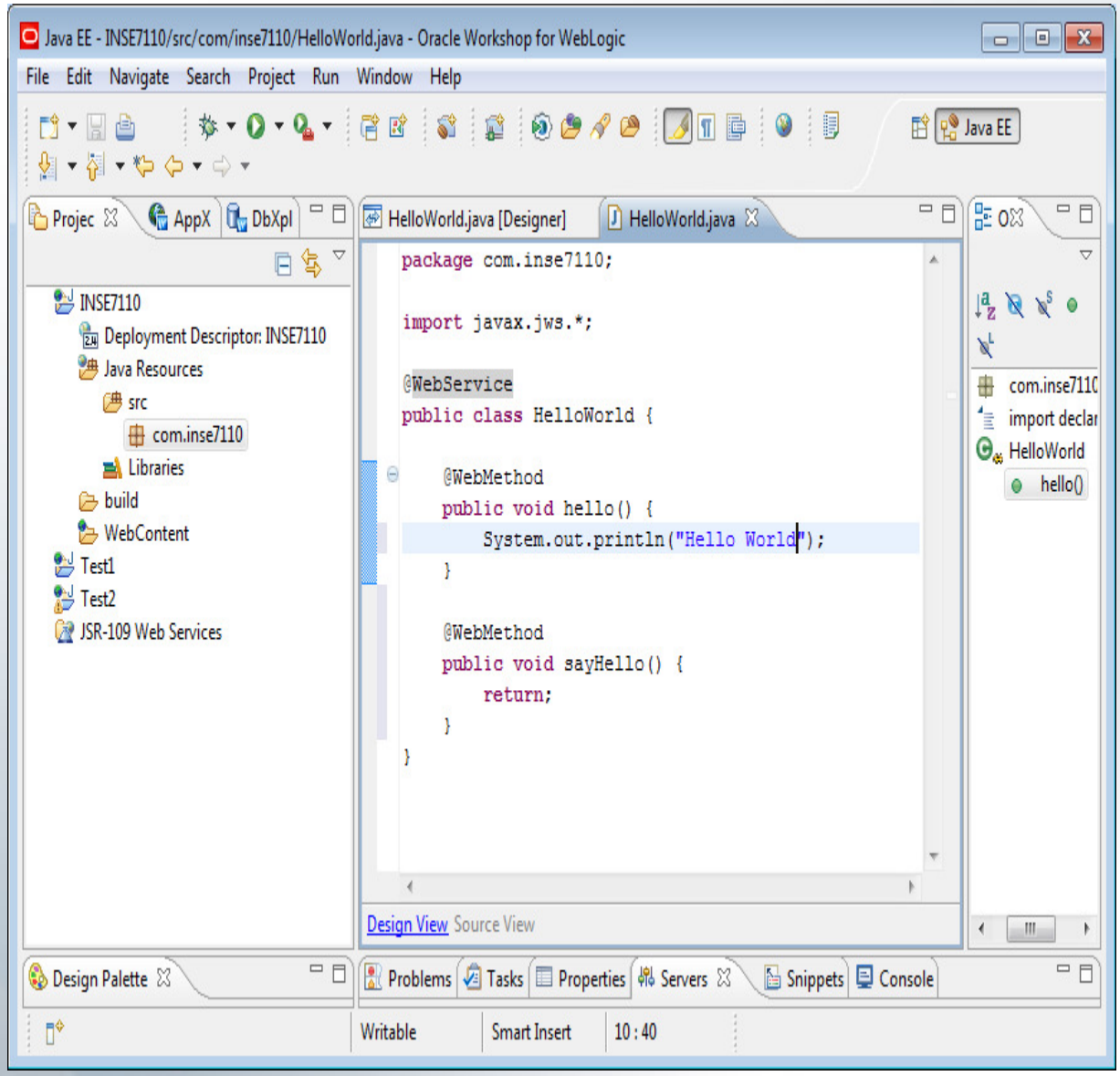
A Concrete Scenario

- Create a Web Service



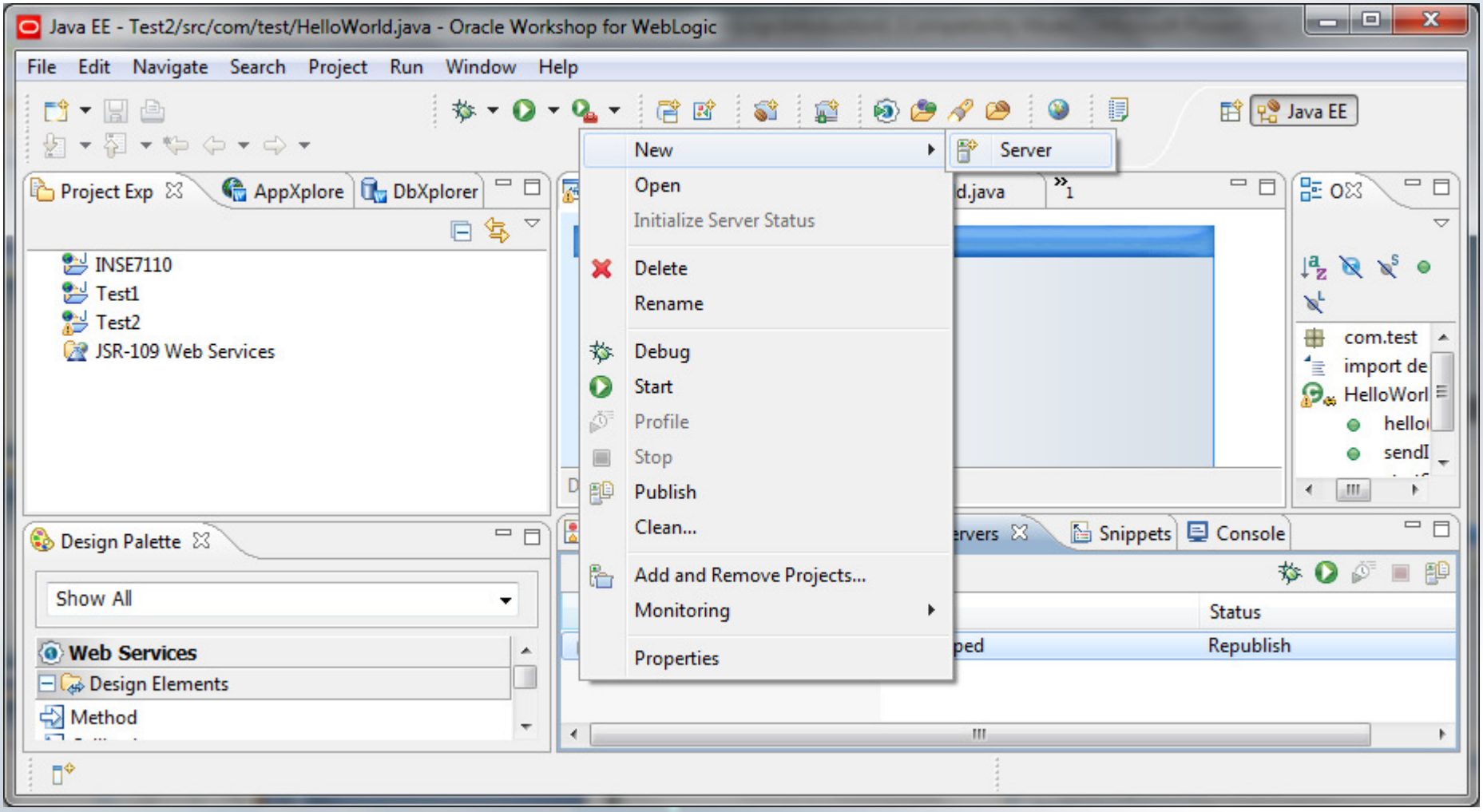
A Concrete Scenario

- Add the Web Service code
  - Implement the business logic of your Web Service
- Using both the Design and Source Views
- Programming language
  - Java



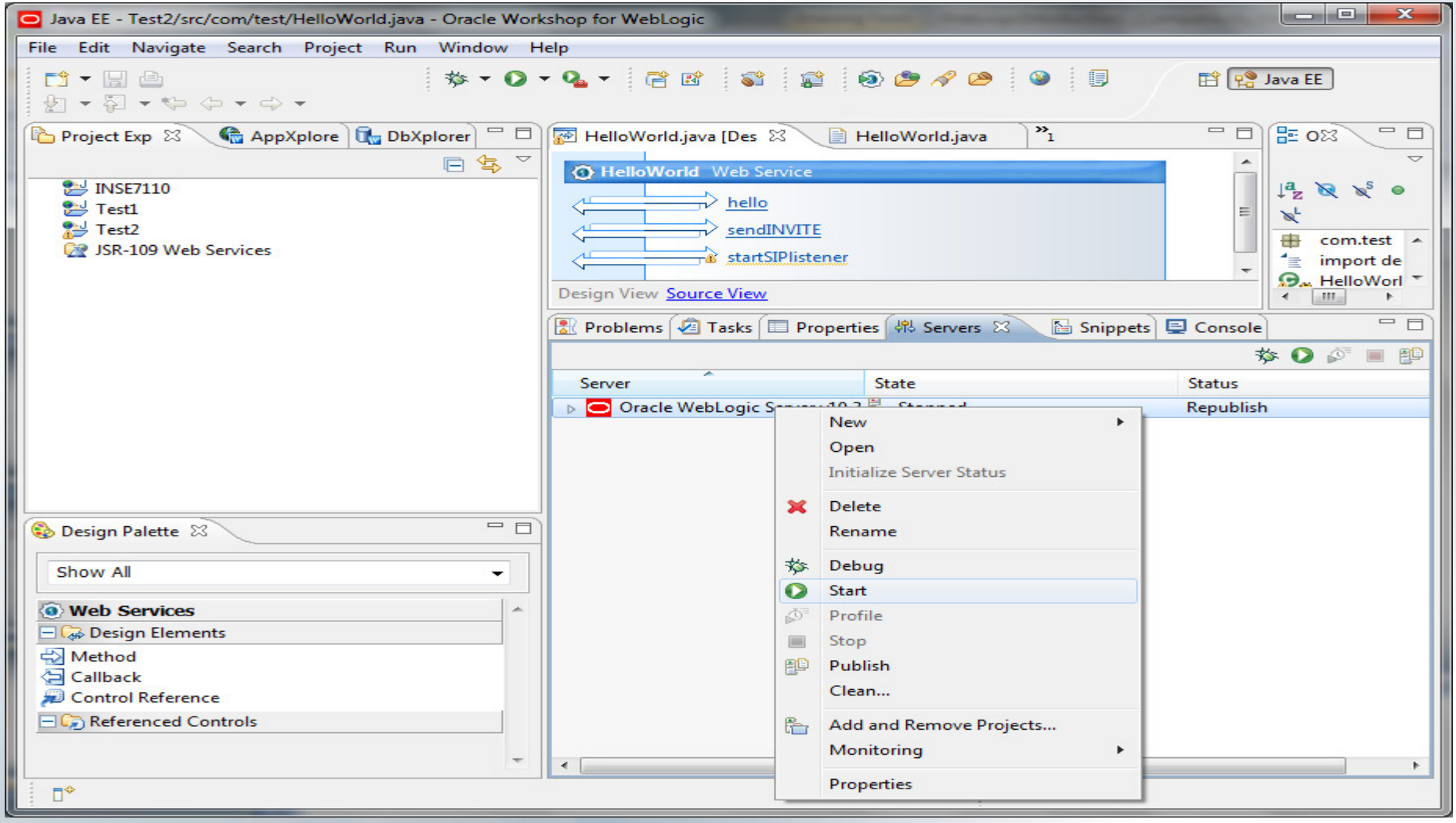
A Concrete Scenario

- Add a weblogic server



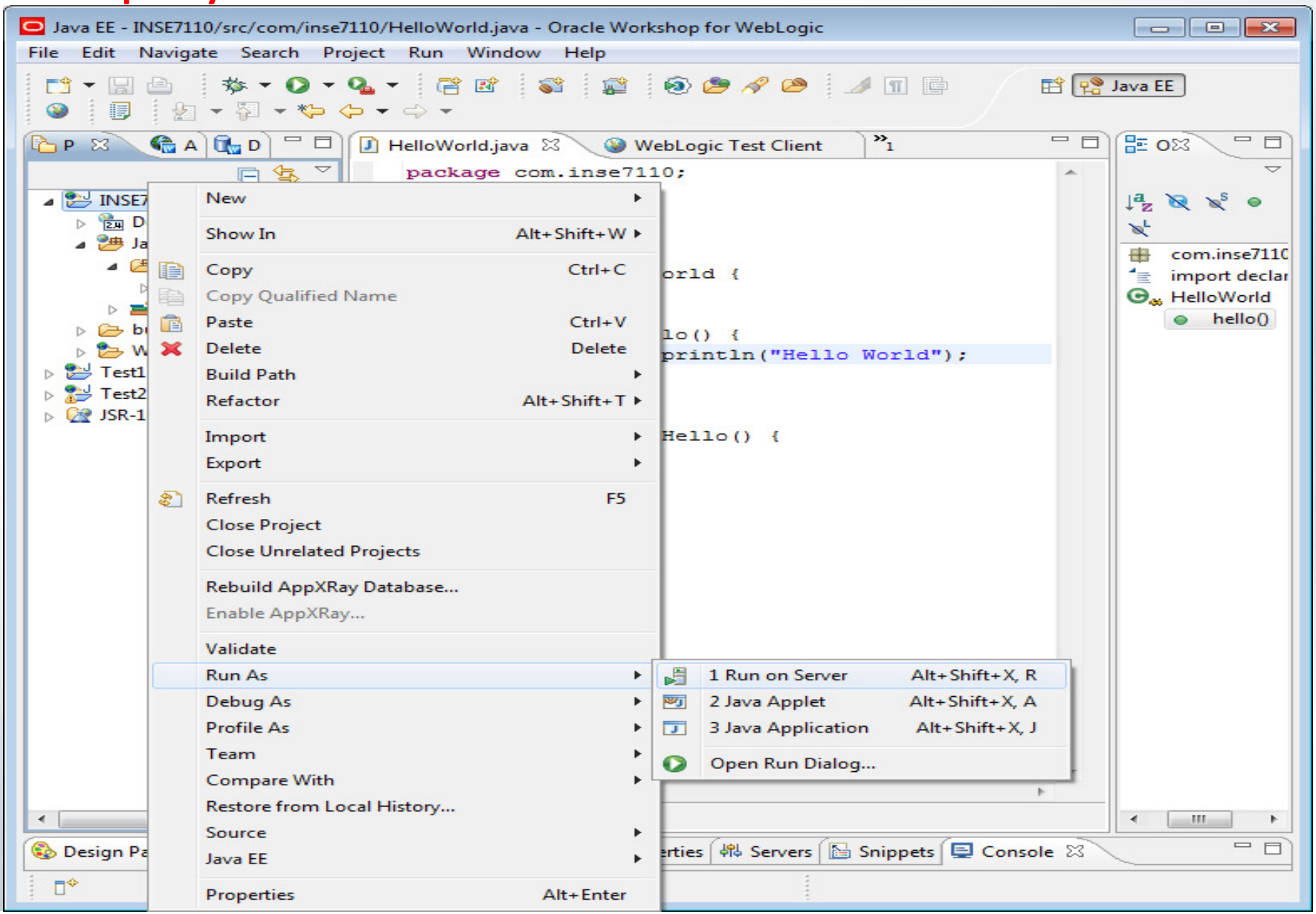
A Concrete Scenario

- Start the weblogic server



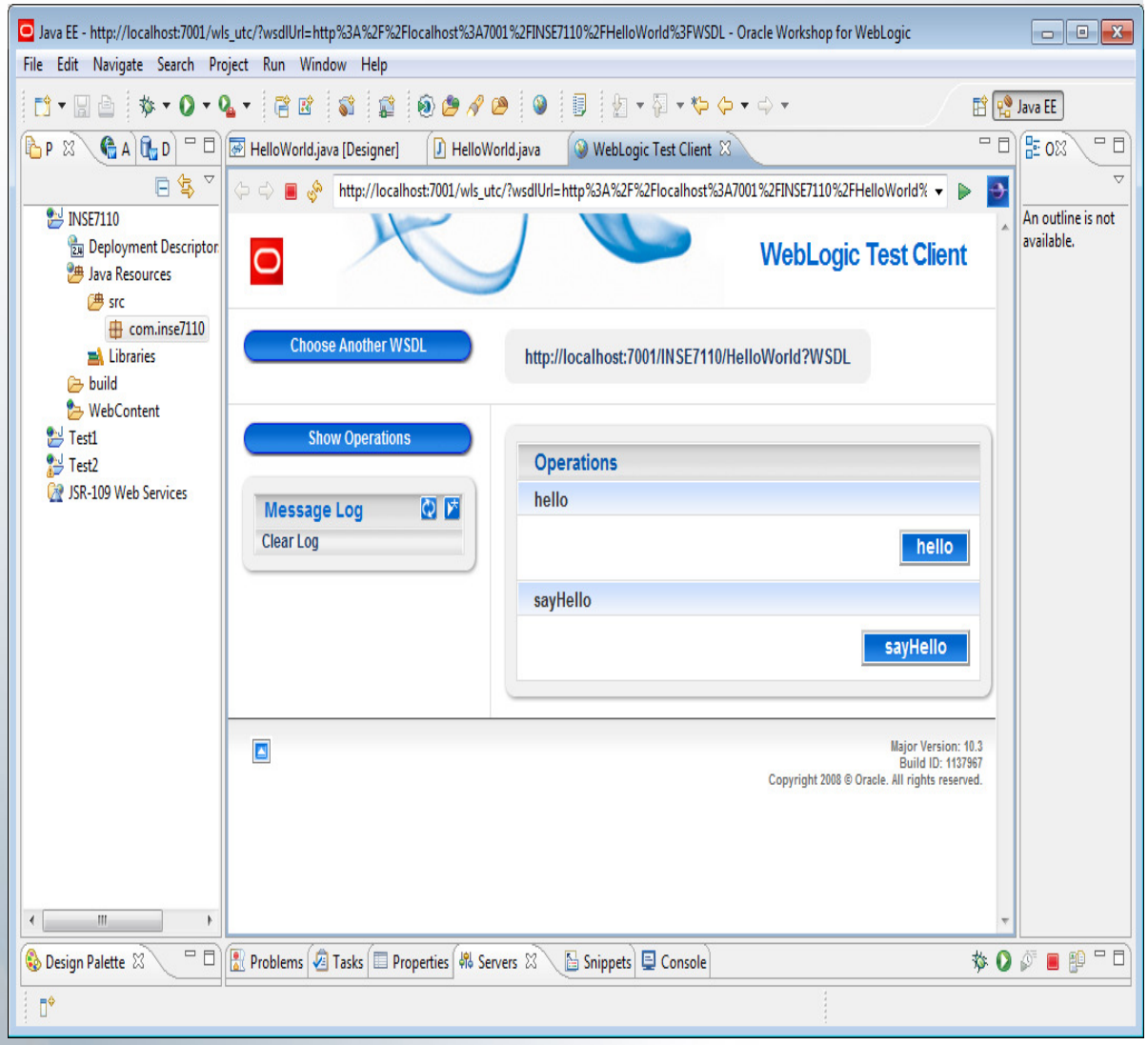
A Concrete Scenario

- Deploy a Web Service



A Concrete Scenario

- Test a Web Service
  - Using Test View
- Test View
  - Invoke a web service method from a browser
  - View the XML messages that are exchanged



Java EE - http://localhost:7001/wls_utc/?wsdlUrl=http%3A%2F%2Flocalhost%3A7001%2FINSE7110%2FHelloWorld%3FWSDL - Oracle Workshop for WebLogic

File Edit Navigate Search Project Run Window Help

Java EE

P A D

- INSE7110
  - Deployment Descriptor
  - Java Resources
    - src
      - com.inse7110
    - Libraries
  - build
  - WebContent
- Test1
- Test2
- JSR-109 Web Services

HelloWorld.java [Designer] | HelloWorld.java | WebLogic Test Client

http://localhost:7001/wls_utc/callOperation.do;jsessionid=xkMjLc1GycT2YwLt8nzWw5mjnBQPpbYH3pP3

Choose Another WSDL

http://localhost:7001/INSE7110/HelloWorld?WSDL

Show Operations

Message Log

hello

Clear Log

hello Request Summary

Arguments:	[void]
Returned:	[void]
Submitted:	Sun Jan 24 20:40:16 EST 2010
Duration:	83 ms

hello Request Detail

Service Request

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
 <env:Header />
  <env:Body>
  <hello xmlns="http://com/inse7110" />
 </env:Body>
 </env:Envelope>
```

Service Response

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
 <env:Header />
  <env:Body>
  <m:helloResponse xmlns:m="http://com/inse7110" />
 </env:Body>
 </env:Envelope>
```

O

An outline is not available.

Design Palette | Problems | Tasks | Properties | Servers | Snippets | Console

Done

A Concrete Scenario

– View the WSDL file

Java EE - http://localhost:7001/wls_utc/?wsdlUrl=http%3A%2F%2Flocalhost%3A7001%2FINSE7110%2FHelloWorld%3FWSDL - Oracle Workshop for WebLogic

File Edit Navigate Search Project Run Window Help

Proj App DbX

SE7110
Deployment Descriptor: INSE7110
Java Resources
src
com.inse7110
Libraries
build
assembly
classes
jws
classes
weboutput
WEB-INF
classes
com
META-INF
policies
HelloWorldService.wsdl
HelloWorldService.xml
HelloWorldService-anno
web.xml
weblogic.xml
weblogic-webservices.xr
weblogic-webservices-p
webservices.xml
weboutput
WEB-INF
web.xml
weblogic.xml
WebContent

HelloWorld.java [Desi | HelloWorld.java | http://localhost:7001 | HelloWorldService.wsd

http://localhost:7001/INSE7110/HelloWorld?WSDL

```
<?xml version="1.0" encoding="UTF-8" ?>
- <s0:definitions name="HelloWorldServiceDefinitions"
    targetNamespace="http://com/inse7110" xmlns=""
    xmlns:s0="http://schemas.xmlsoap.org/wsdl/" xmlns:s1="http://com/inse7110"
    xmlns:s2="http://schemas.xmlsoap.org/wsdl/soap/">
  - <s0:types>
    - <xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
        targetNamespace="http://com/inse7110"
        xmlns:s0="http://schemas.xmlsoap.org/wsdl/"
        xmlns:s1="http://com/inse7110"
        xmlns:s2="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:xs="http://www.w3.org/2001/XMLSchema">
      - <xs:element name="sayHello">
        - <xs:complexType>
            <xs:sequence />
          </xs:complexType>
        </xs:element>
      - <xs:element name="sayHelloResponse">
        - <xs:complexType>
            <xs:sequence />
          </xs:complexType>
        </xs:element>
      - <xs:element name="hello">
        - <xs:complexType>
            <xs:sequence />
          </xs:complexType>
        </xs:element>
      - <xs:element name="helloResponse">
        - <xs:complexType>
            <xs:sequence />
          </xs:complexType>
        </xs:element>
```

An outline is not available.

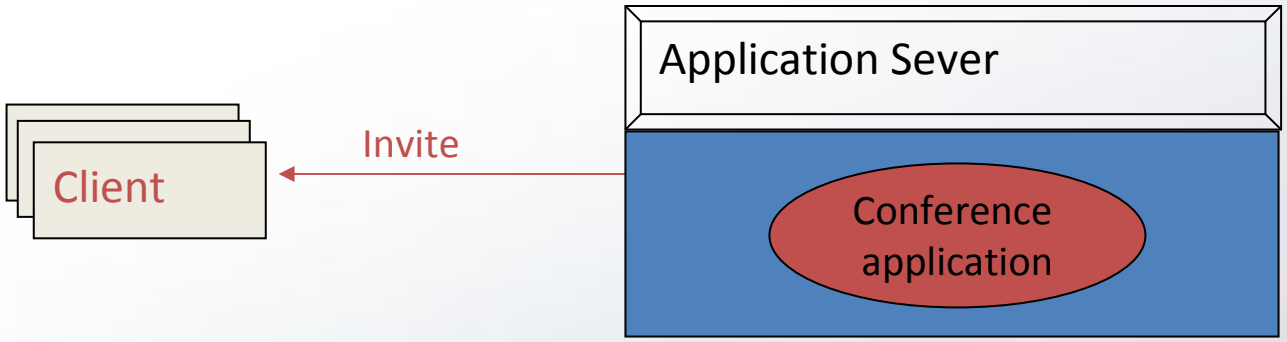
Design Palette

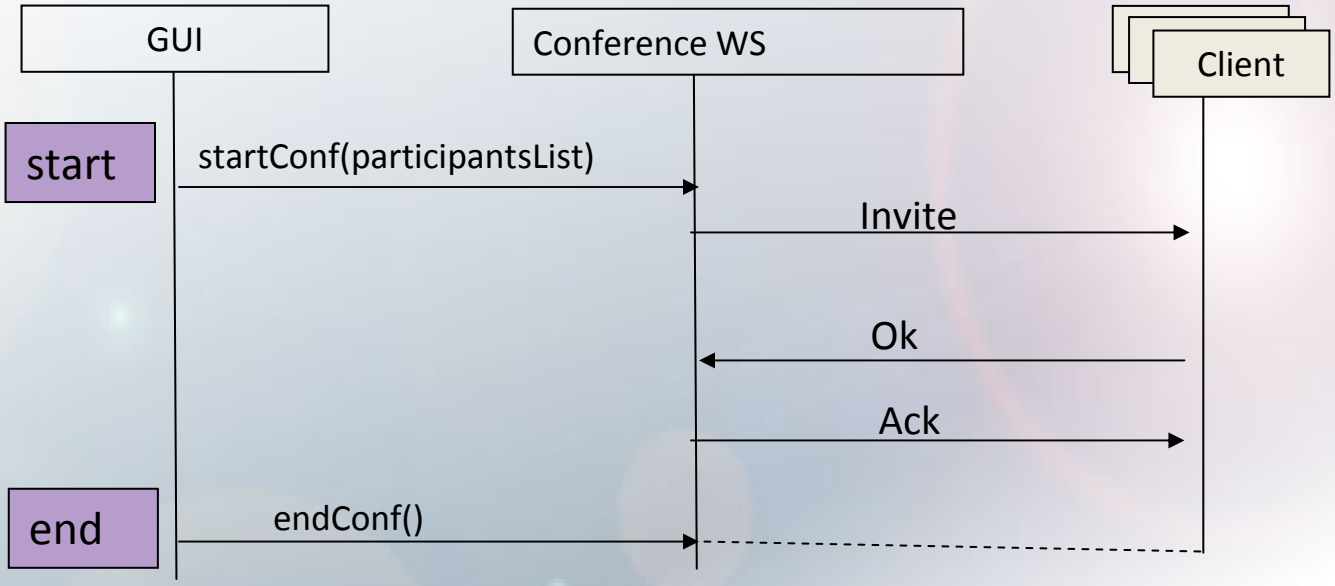
Problems Tasks Properties Servers Snippets Console

Done

Hints for the project

What to implement





Hints for the project

- Server side:
  - a conferencing web service that implements a simplified conference focus
    - uses SIP stack to listen to, receive, process and send SIP messages
      - Use JAIN SIP
- Client side: SIP client
  - Download or
  - Code your client using SIP stack
- Use of JMF for media

How to download JAIN SIP

- JSR 32 Jain SIP
- API doc and .jar can be downloaded at:
  - http://jcp.org/aboutJava/communityprocess/mrel/jsr032/index.html
  - Unzip
- Reference implementation .jar can be downloaded at:
  - http://download.java.net/communications/jain-sip/nightly/
  - Click: “jain-sip-ri/”, you can find all nightly build implementations .jar
  - You can also find sdp implementations there…

- When implementing a web service using Jain SIP, put the two jar files (one api, one impl) into your project: \WEB-INF\lib
  - jsip_api_v1.2.jar
  - jain-sip-ri-1.2.xx.jar

- Tip: you may need log4j-xxx.jar if you always have running errors when using Jain SIP (this depends on which sip impl that you use)
  - http://www.apache.org/dyn/closer.cgi/logging/log4j/1.2.15/apache-log4j-1.2.15.zip

A brief introduction to JAIN SIP

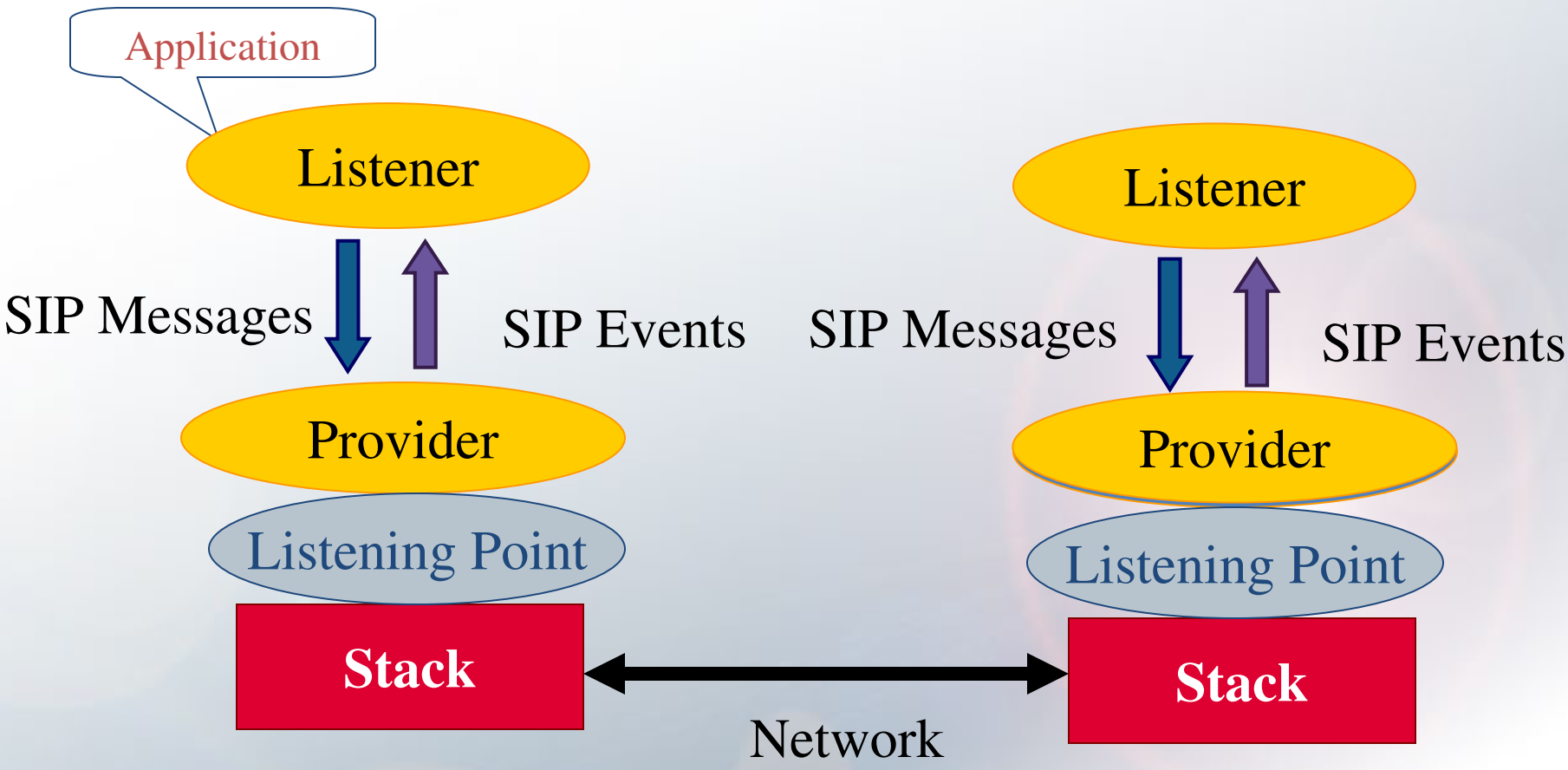
- JAIN-SIP is a java-standard interface to a SIP signaling stack
- It provides a low level Java API specification for SIP Signaling
- It was designed for the developers who require fine grained access to the SIP protocol
- It can be used in:
  - Clients
  - Servers
  - Proxies
- JAIN SIP reference implementation is open source, very stable, and very widely used

- Introduction to JAIN SIP
  - http://www.oracle.com/technology/pub/articles/dev2arch/2007/10/introduction-jain-sip.html

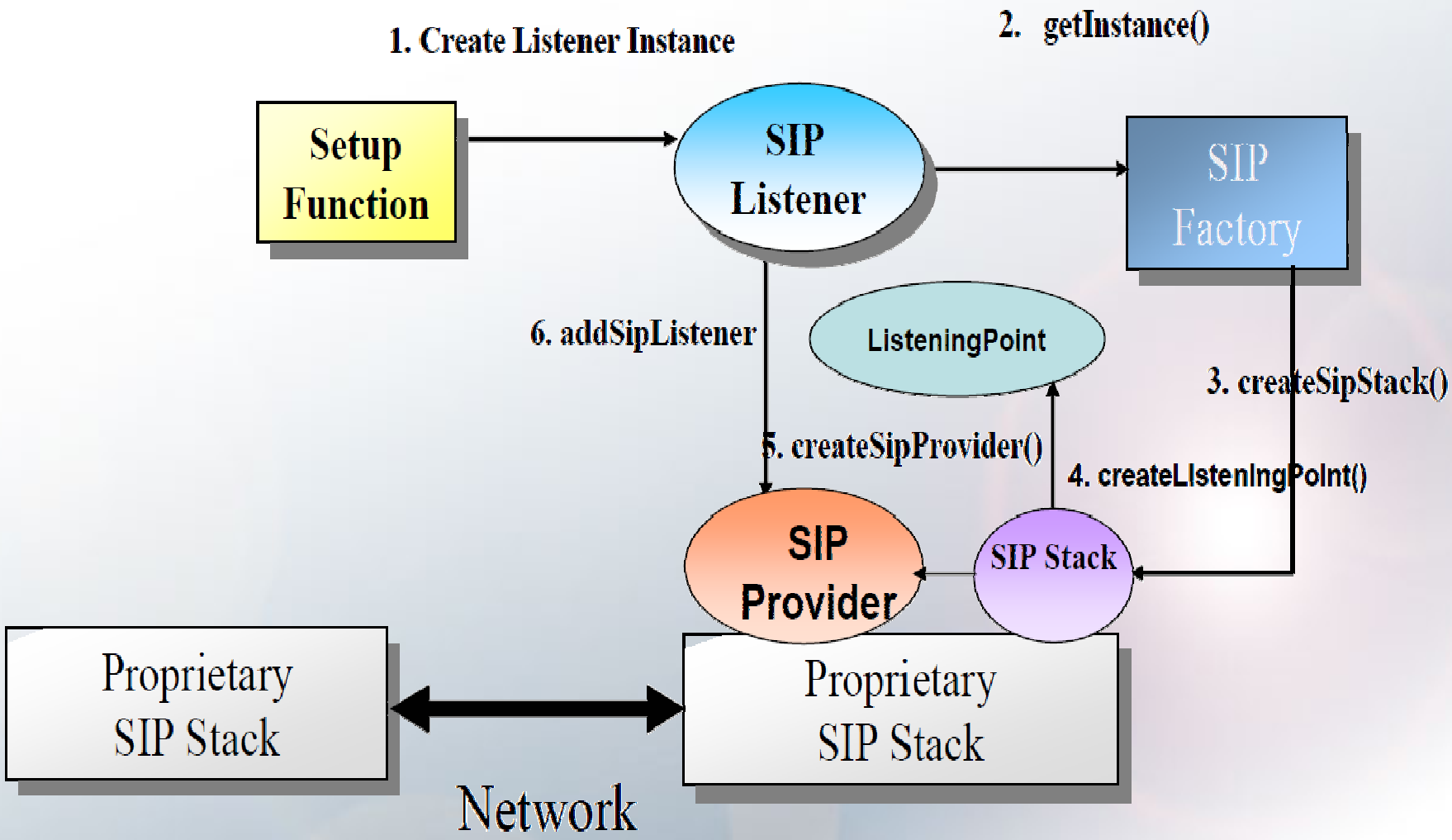
A brief introduction to JAIN SIP

- JAIN SIP services
  - Provides methods to format and send SIP messages
  - Parses incoming messages and enable the application to access the messages' fields via a standardized JAVA interface
  - Invokes appropriate application handlers when appropriate (message arrivals, transaction time-outs)
  - Provide Transaction support and manage Transaction state and lifetime on behalf of a user application.
  - Provide Dialog support and manage Dialog state and lifetime on behalf on a user application

JAIN SIP Architecture



A brief introduction to JAIN SIP



A brief introduction to JAIN SIP

Basic steps:

1. **public class** MySipListener **implements** javax.sip.SipListener {…}
2. SipFactory sipFactory = SipFactory.*getInstance*();
3. SipStack sipStack = sipFactory.createSipStack(properties);
4. ListeningPoint listeningPoint = sipStack.createListeningPoint(address, port, transport);
5. SipProvider sipProvider = sipStack.createSipProvider(listeningPoint);
6. sipProvider.addSipListener(mySipListener);

- Introduction to JAIN SIP
  - http://www.oracle.com/technology/pub/articles/dev2arch/2007/10/introduction-jain-sip.html

SDS or Web Logic ?

- Ericsson SDS:
  - Need to configure IMS first…
  - Can use SIP Servlet
  - Trigger: from SIP client or HTTP Servlet
    - Both need programming
  - Conference client registration: register to CSCF

- Oracle Web Logic
  - No configuration is needed
  - Can use SIP Stack (lower level of abstraction than SIP Servlet)
  - Trigger: from web logic test client or any web service that talks SOAP
    - The first option is provided by the tool, no need to code
  - Conference client registration: register to your SIP Listener (in your WS logic)

- What's in common:
  - Media handling: RTP/RTCP (JMF)
  - Conference client: SIP client that talks RTP

References

- http://download.oracle.com/docs/cd/E12840_01/wls/docs103/intro/chap1.html
- http://download.oracle.com/docs/cd/E12840_01/wls/docs103/webserv_intro/overview.html#choose
- http://download.oracle.com/docs/cd/E12840_01/wls/docs103/webserv_intro/standards.html#wp1078494
- http://www.oracle.com/technology/products/workshop/index.html
- http://e-docs.bea.com/wlw/docs103/index.html
- http://e-docs.bea.com/wls/docs103/webservices.html
- http://www.oracle.com/technology/pub/articles/dev2arch/2007/10/introduction-jain-sip.html

Q&A