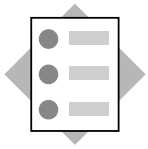
Using Web Services as Web Dynpro Model

Contents:

- **Web Services overview**
- **Implementing a Web Service Client**

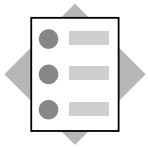
Using Web Services as Web Dynpro Model: Objectives



After completing this lesson, you will be able to:

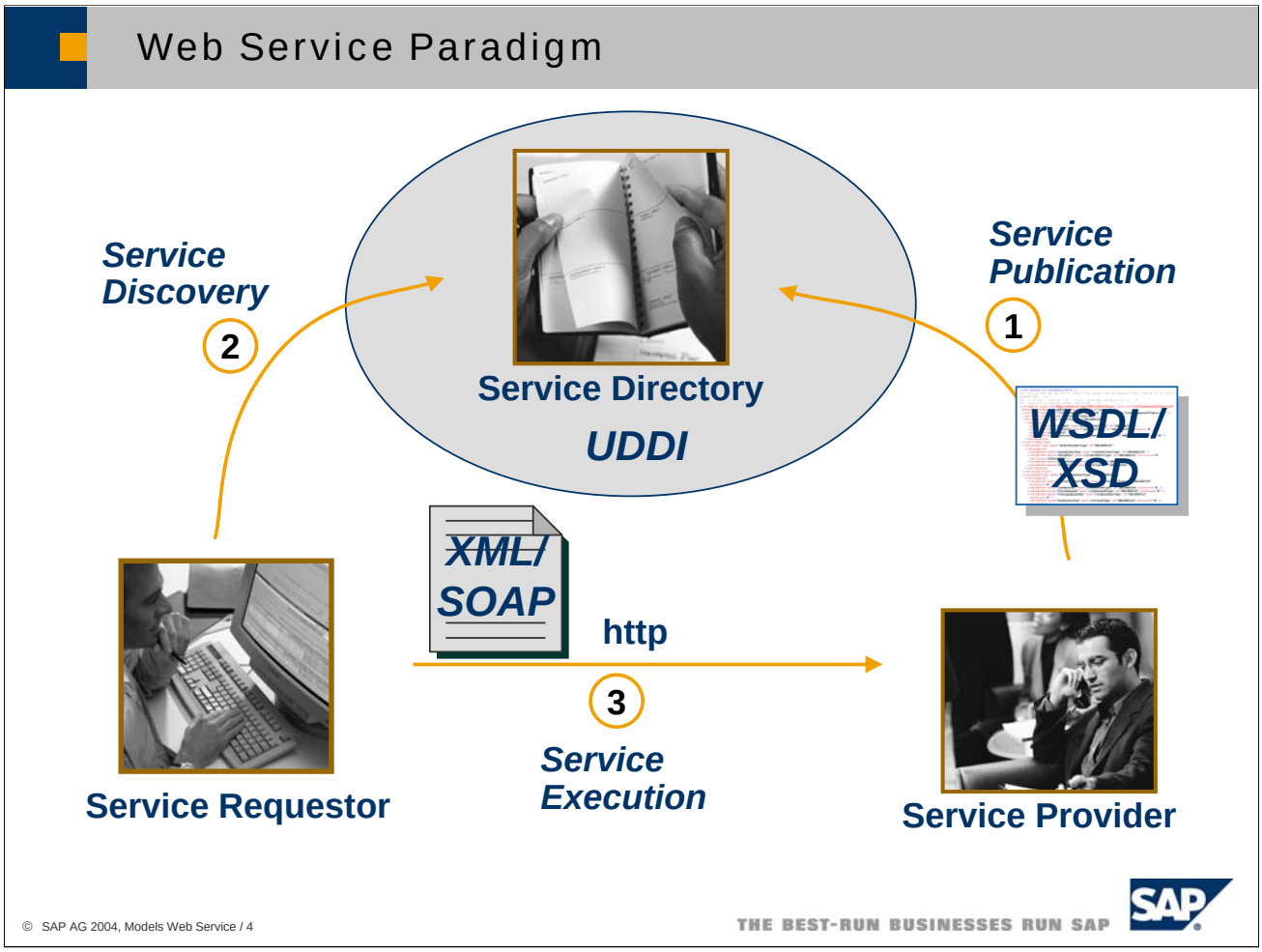
- **Explain what Web Services are.**
- **Implement a simple Web Dynpro application using a Web Service.**

Web Services Overview: Topic Objectives



After completing this topic, you will be able to:

- **Explain what Web Services are.**



- **Web Services Paradigm**
  Web Services are appearing on the Internet in the form of e-business sites and portal sites. For example, *priceline.com* and *expedia.com* act as the broker for airlines, hotel and car booking respectively. They are statically composed web services that have pre-negotiated understanding with certain airlines and hotels and broker their services through their portal sites. These are mostly B2C kind of web services. A large number of technologies and platforms are appearing and are being standardized upon so as to enable the paradigm of web services, for satisfying B2B and B2C scenarios alike in a uniform manner. These standards and platforms enable creation and deployment, description, discovery and communication amongst them.
  Web Services Description Language (WSDL) is used to publish services' access points (i.e., bindings) and supported interfaces, both of which are described in an XML-based description language.
  UDDI is used for registration and description of web services. After having discovered its partners, web-services use the document model to asynchronously exchange documents, and **Simple Object Access Protocol (SOAP)** for messaging (which is an incarnation of remote procedure call (RPC) in eXtensible markup language (XML)) over hypertext transfer protocol (HTTP).
  Most services are implemented using platform independent languages such as Java and C# on platforms like J2EE and .Net. The primary means for enforcing security are digital signature and strong encryption with public/private key pairs. Standards like SAML and XKMS are appearing in this area. A large number of payment mechanisms are being defined, too.

WSDL Overview

Web Service Description Language

Describes the basic characteristics of a Web service

- **Supported operations and their data format** **e.g. xCBL Order**
- **Supported protocols** **e.g. SOAP**
- **Network address** **e.g. http://a.com/order**

Further information

- **Working Group:** **http://www.w3.org/2002/ws/desc/**
- **Specification:** **http://www.w3.org/TR/wsdl12/**

- **Web Services Description**
  In traditional software development environment, software component interfaces are defined through interface definition languages (IDL). The interfaces describe the operations the software component supports, their inputs and the expected outputs. This enables the interfaces to be decoupled from the actual implementation. As web services are envisaged as software available on the web that other web services or users will use, they need to be described so that other components can easily use them without coupling the interfaces with the implementations.

WSDL Structure (simplified)

```
<?xml version="1.0" encoding="utf-8" ?>
<definitions>
 <types>
    … <element name="qty" type="string"
                minOccurs="0"/> …
 </types>

 <message name="POMessageIn">
   … <part name="Quantity" type="qty"/> …
 </message>

 <portType name="POPortType">
  <operation>
   … <input message="POMessageIn" /> …
  </operation>
 </portType>

 <binding name="SOAP" portType="POPortType">
  … SOAP/HTTP binding definition …
 </binding>
 <service name="OrderWineService">
   <port name="Order" binding="SOAP">
    <address
      location="http://www.dijan.fr/Order/"/>
   </port>
 </service>
</definitions>
```

What

- A *portType* describes the abstract interface (Web service type) of the Web service
- Each contained *operation* can have an *input*, an *output* and a number of *fault messages*
- Different Messages are build from build-in or custom data types
- Data types are defined using XML Schema

How

- A *binding* specifies exactly one protocol for the operations of a portType

Where

- A *port* defines the Web service endpoint by specifying a single network address

- **Web Services Description Language (WSDL)**
  WSDL is an attempt to describe the web service interfaces. WSDL enables creation of flexible and generic interaction models for web services. WSDL enables description of web services irrespective of the message formats and network protocols used. For example, in WSDL a service is described through a set of endpoints. An endpoint is in turn a set of operations. An operation is defined in terms of messages received or sent out by the web service:

  - Message – an abstract definition of data being communicated consisting of message parts,
  - Operation – an abstract definition of an action supported by the service. Operations are of the following type namely, one-way, request-response, solicit-response, notification,
  - Port type – an abstract set of operations supported by one or more end points,
  - Binding – a concrete protocol and data format specification for a particular port type,
  - Port – a single end point defined as a combination of a binding and a network address,
  - Service – a collection of related end-points.

  As the implementation of the service changes or evolves in time, the WSDL definitions have to be continuously updated and versioning of the descriptions have to be tracked.

UDDI Overview

Universal Description, Discovery and Integration (UDDI)

Describes how to advertise and discover a Web service

- **Differentiates Web service provider, Web service and Web service type**
- **Holds metadata that can be used to search for services (names, IDs, categories, types, etc.)**
- **Specifies the interface for Web service registries**

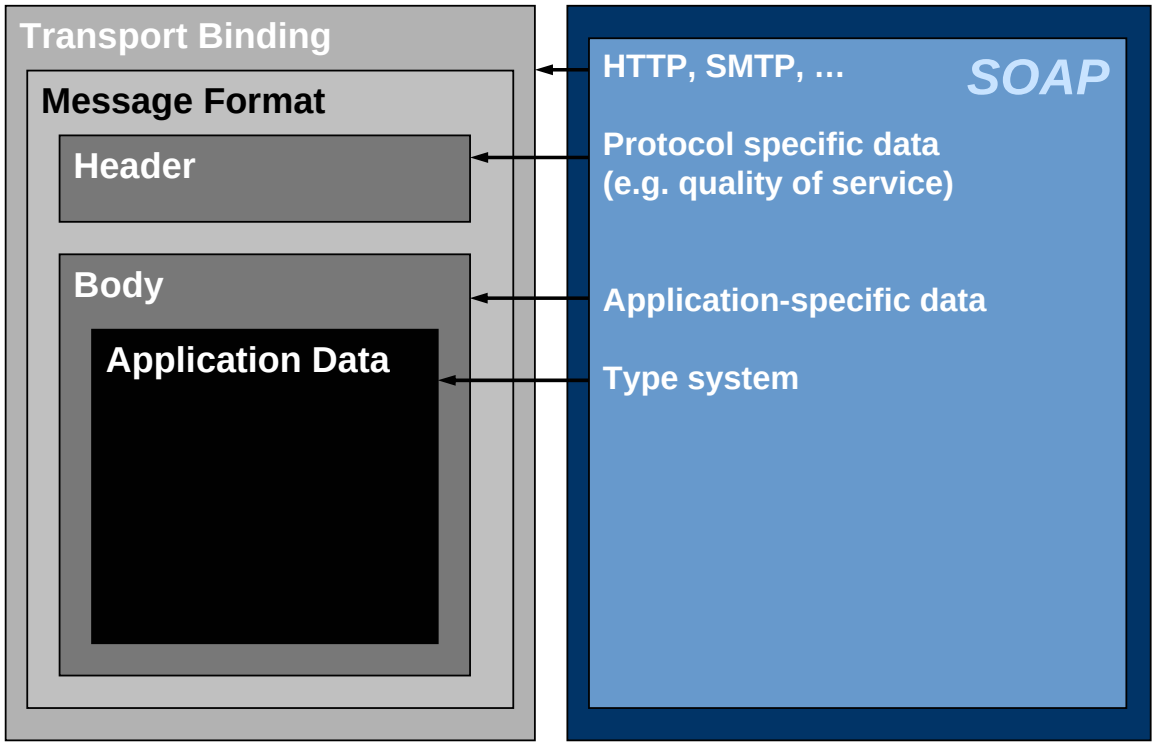
UDDI Business Registry

- ***THE* directory for Web services on the Internet**
- **Publicly available, free of charge**
- **Operated by SAP, IBM, Microsoft, and NTT Communications**
- **UDDI Version 3 Beta available now**
- **SAP's node at http://uddi.sap.com**

- **Web Services Discovery**

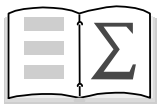
  While web browsing for information, search-engines are used to find relevant web sites based on keywords. However, this leads to lot of unnecessary links that need to be sifted through, before arriving at the relevant sites. Similarly, before web services can interact with each other, they need to discover other compatible web services with which they can undertake business. The registration and discovery of web services necessitate other entities that act as intermediaries. Universal Description Discovery Integration (UDDI) is such an initiative. It is a group of web-based registries (operator sites) that expose information about a business and its technical interfaces and APIs. The core component of the UDDI is the registration, an XML file used to define business and the web services they provide. There are three parts to the registration, namely *white pages* for name, address, contact and other identifiers, *yellow pages* for classification of business under standardized taxonomies and *green pages* that contain technical information about the web services that are exposed. It also exposes a set of APIs for inquiry and publication of information related to web services. The inquiry APIs enable browsing of the information in the repository site (e.g. find_business) and also to drill down (e.g. get_businessDetail). The publication APIs are for publishers to put their information on these repositories.
  In the SAP Web AS, UDDI client and server functions are provided. You can search in all, and publish to all, registries that conform to the standard. A full-blown UDDI server is shipped as part of the SAP Web AS so customers can create their own registries. SAP also provides a public UDDI Business Registry under uddi.sap.com.

SOAP Structure and Features



- **SOAP**
  SOAP defines a common layer for document exchange. Services can define their own service-specific content on
  the top of SOAP.

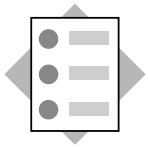
Web Services Overview: Topic Summary



You should now be able to:

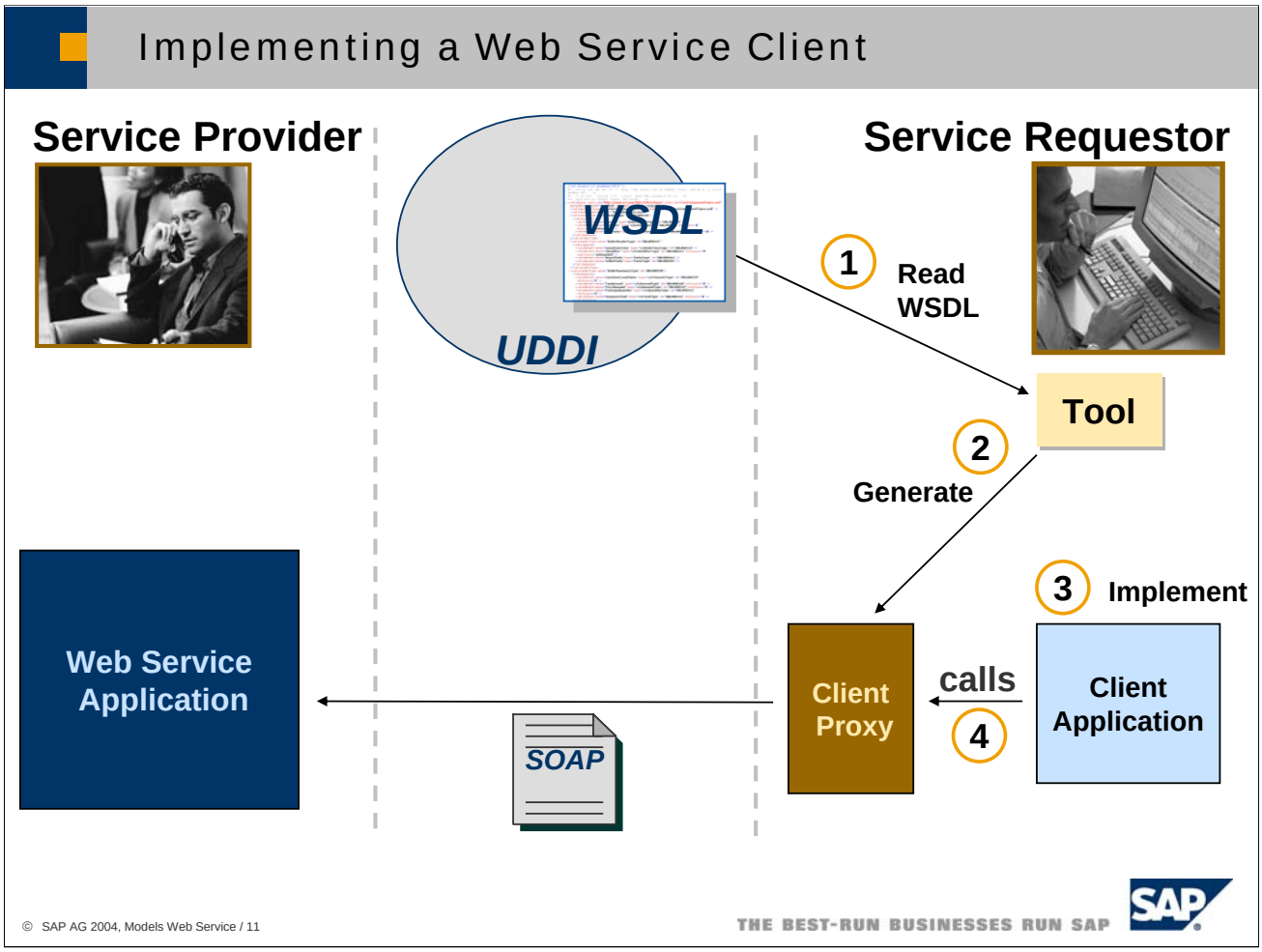
- **Explain what Web Services are.**

Implementing a Web Service Client: Topic Objectives



After completing this topic, you will be able to:

- **Implement a Web Service client.**



■ **Implementing a Web Service Client**

Step 1: Retrieve description of Web Service
Retrieval of URL of WSDL description

- either manually,
- or using UDDI Browsing

Supported by the Web Service Infrastructure via

- Web Service Proxy Project
- UDDI Client Browser

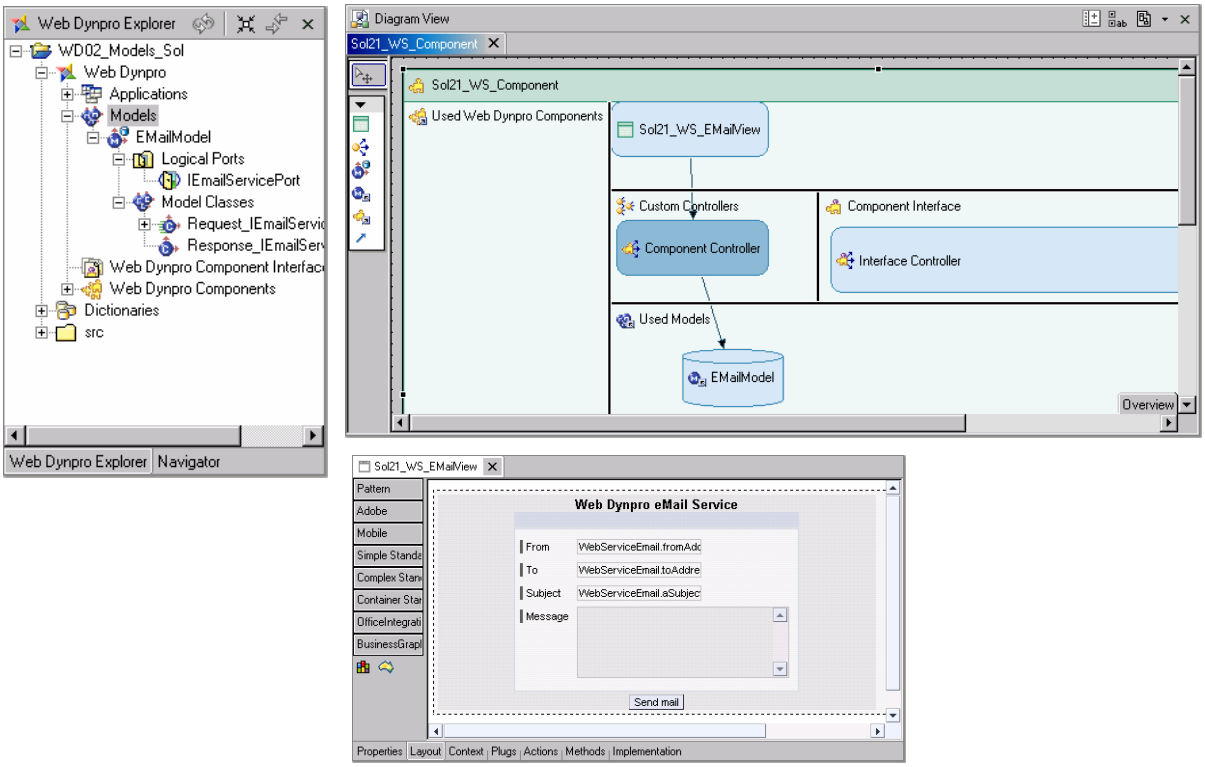
Step 2: Generate Web Service Client Proxy
Start proxy generator based on imported WSDL Document
Supported by the Web Service Infrastructure via

- Web Service Proxy Project

Step 3: Implementation of Client Application
Use generated client proxies
Supported by the Web Service Infrastructure via

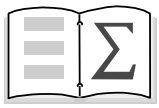
- the standard Java IDE

Web Dynpro and Web Services



- **Web Dynpro and Web Services**
  With the introduction of Web service as a possible model type, Web Dynpro now offers a much more comprehensive set of choices for connecting a Web Dynpro front end to an existing backend system. The developer can now point the model wizard to any Web Service on the Internet and have the code for accessing this Web Service generated automatically.

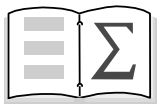
Implementing a Web Service Client: Topic Summary



You should now be able to:

- **Implement a Web Service client.**

Using Web Services as Web Dynpro Model: Unit Summary



You should now be able to:

- **Explain what Web Services are.**
- **Implement a simple Web Dynpro application using a Web Service.**