

## Chapter - Sending and receiving complex data structures- Referring existing elements

### Objectives

---

- At the end of this chapter you will be able to understand
  - ♦ How to send and receive complex data to and from a web service, which has been generated by consuming third party schemas

## Referring to existing XML elements

- For the moment we're defining XML elements such as `<productQuery>` directly in the WSDL file.
- However, in practice, most likely such elements are defined by a 3rd party such as an industrial consortium or neutral association.
- Suppose that they are provided in a file *purchasing.xsd* such as this:



3

## Referring to existing XML elements

The root element is `<schema>`

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://bar.org/purchasing"
  xmlns:tns="http://bar.org/purchasing">
  <element name="productQuery">
    <complexType>
      <sequence>
        <element name="queryItem" minOccurs="1"
          maxOccurs="unbounded" type="tns:queryItemType"/>
      </sequence>
    </complexType>
  </element>
  <element name="productQueryResult">
    <complexType>
      <sequence>
        <element name="resultItem" maxOccurs="unbounded"
          minOccurs="1">
          <complexType>
            <attribute name="productId" type="string"/>
            </complexType>
          </element>
        </sequence>
      </complexType>
    </element>
    <complexType name="queryItemType">
      <attribute name="productId" type="string"/>
      <attribute name="qty" type="int"/>
    </complexType>
    <element name="invalidProductId" type="string"/>
    <element name="invalidQty" type="int"/>
  </schema>
```

The default namespace is the XML schema namespace, so you don't need to use the xsd prefix below.

As they are defined by a 3<sup>rd</sup> party, it should use a different target namespace. Let's assume that it is `http://bar.com/purchasing`.

Everything else remains unchanged



4

## Referring to existing XML elements

---

- How to refer to those XML elements in our WSDL file?

5



## Referring to existing XML elements

---

- First, put the *purchasing.xsd* file into the same folder as the WSDL file (i.e., the project root).
- Then modify the WSDL file:

6



# Referring to existing XML elements- wsdl

You're saying: I'd like to refer to the XML elements defined in the `http://bar.org/purchasing` namespace. Then the XML elements will be visible to this WSDL file. This is like the import statement in Java used to import a package or a class

You don't need to define your own elements anymore

How can the WSDL parser find out the XML elements defined there? It will work if the person parsing the WSDL have set up a table like below. Such a table is called an XML catalog.

Namespace	Path to its xsd file
<code>http://bar.org/purchasing</code>	<code>c:\schema\fl xsd</code>
<code>http://...</code>	<code>c\...</code>
<code>http://...</code>	<code>c\...</code>

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://foo.com"
  xmlns:p="http://bar.org/purchasing"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  name="BizService"
  targetNamespace="http://foo.com">
  <wsdl:types>
    <xsd:schema targetNamespace="http://foo.com"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:import
        namespace="http://bar.org/purchasing"
        schemaLocation="purchasing.xsd"/>
      <xsd:element
        name="productQuery"/>
    </xsd:schema>
  </wsdl:types>
  <wsdl:message name="queryRequest">
    <wsdl:part name="parameters" element="p:productQuery" />
  </wsdl:message>
  <wsdl:message name="queryResponse">
    <wsdl:part name="parameters" element="p:productQueryResult" />
  </wsdl:message>
  <wsdl:message name="queryInvalidProductId">
    <wsdl:part name="NewPart" element="p:invalidProductId" />
  </wsdl:message>
  <wsdl:message name="queryInvalidQty">
    <wsdl:part name="NewPart" element="p:invalidQty" />
  </wsdl:message>
  <wsdl:portType name="BizService">
    <wsdl:portType>
    <wsdl:binding name="BizServiceSOAP" type="tns:BizService">
    </wsdl:binding>
    <wsdl:service name="BizService">
    </wsdl:service>
  </wsdl:definitions>
```

As you'll be giving away this WSDL to many people, it may be too difficult to ask everyone to set up the XML catalog. So you may simply distribute the XSD file and make sure it is in the same folder as the WSDL file and specify the relative path here. In addition to the XML catalog, their WSDL processor will follow this path to find the XSD file.

The elements are now defined in another namespace



7

## Referring to existing XML elements

- Modify build.xml:



8

## Referring to existing XML elements- build.xml

```
<project ...>
...
<target name="generate-service">
  <wsdl2code
    wsdlfilename="${name}.wsdl"
    serverside="true"
    generateservicexml="true"
    skipbuildxml="true"
    serversideinterface="true"
    namespacepackages=
      "http://foo.com=com.ttdev.biz,http://bar.org/purchasing=com.ttdev.biz.purchasing"
    targetsourcefolderlocation="src"
    targetresourcesfolderlocation="src/META-INF"
    overwrite="true"/>
  <replaceregexp
    file="src/META-INF/services.xml"
    match="${name}Skeleton"
    replace="${name}Impl"/>
  </target>
<target name="generate-client">
  <wsdl2code
    wsdlfilename="${name}.wsdl"
    skipbuildxml="true"
    namespacepackages="http://foo.com=com.ttdev.biz.client"
    targetsourcefolderlocation="src"
    overwrite="true"/>
  </target>
</project>
```

As the XML elements are in the  
http://bar.org/purchasing namespace,  
you may want to map it to a Java  
package.

Separate them by  
a comma

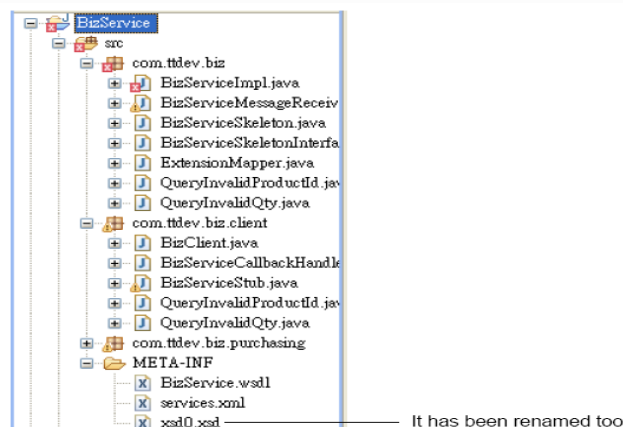
You could do the same thing for the  
client, but by default the XML elements  
will be mapped to inner classes of the  
client stub. So you don't need to specify  
a package for them.

9

**seed**  
beyond the obvious

## Referring to existing XML elements

- Run the build and we will be able to see following generated files.



10

**seed**  
beyond the obvious

## Quick Recap . . .

---

- We can generate the web services based on a wsdl which import the elements from third party