Enabling Grids for E-science in Europe

www.eu-egee.org

17th October 2004

WSDL

Using <import>

EGEE is a project funded by the European Union under contract IST-2003-508833

Goals

- To examine the uses of the import element in WSDL documents
- To see how the use of this element can contribute to re-use and maintainability in WSDL documents.

One document or many

- WSDL documents are often thought of and created as single monolithic entities.

- In order to enhance manageability and re-use of WSDL, where it is created manually, the document can be split into sub –documents.

- To do this we use the WSDL `<import>` tag.

<import> element

```
<definitions
   targetNamespace="urn:3950"
   xmlns= "http://schema.xmlsoap.org/wsdl/"
   xmlns:xsd= "http://www.w3c.org/2001/XMLSchema"
   xmlns:soap= "http://schemas.xmlsoap.org/wsdl/soap/"
   xmlnssoapenc= "http://schemas.xmlsoap.org/soap/emcoding/"
   xmlns:tns= "urn:3950">

   <import namespace= "http://nesc.ac.uk" location=
   "http://nesc.ac.uk/ez.xsd"/>
```

Acts like C/C++ #include , or Java import.
Incorporates external namespaces

An example of adding complex data types

- We can examine an example of how to include a complex data type without making our WSDL overly long.

- The example is based around a notional service `book service` which contains an ‘object’ `BookInfo`

- We will split the definitions into two XMLSchema files (*a schema may only contain single* `<schema>` *element*).

Service related definitions *bookTypes.xsd*

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:jwsnut.chapter2.bookservice/types/BookQuery"
………>
   <complexType name="ArrayOfBookInfo">
        <complexContent>
                <restriction base="soap-enc:Array">
                        <attribute ref="soap-enc:arrayType"
   wsdl:arrayType="tns:BookInfo[]"/>
                </restriction>
        </complexConent>
   </complexType>
<complexType name="BookInfo">
   <sequence>
        <element name="author" type="string"/>
        <element name="title" type="string"/>
   </sequence>
</complexType>
</schema>
```

JAX-RPC specific type definitions *baseTypes.xsd*

Enabling Grids for E-science in Europe

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://java.sun.com/jax-rpc-ri/internal" ………>
<complexType name="hashMap">
   <complexContent>
        <extension base="tns:map">
                <sequence/>
        </extension>
   </complexContent>
</complexType>
<complexType name="map">
   <complexContent>
        <restriction base="soap-enc:Array">
                <attribute ref="soap-enc:arrayType"
   wsdl:arrayType="tns:mapEntry[]"/>
        </restriction>
   </complexConent>
</complexType>
<complexType name="mapEntry">
   <sequnece>
        <element name="key" type="anyType"/>
        <element name="value" type="anyType"/>
   </sequnece>
</complexType>
</schema>
```

Comparing the schema

- These are each free standing XMLSchema documents
- Each has its own <schema> element and declares a target namespace for its definitions.
- These namespaces are different.
- `bookTypes.xsd` uses the book service namespace
- `baseTypes.xsd` uses the private JAX-RPC reference implementation namespace.

Using these schema in WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
   <definitions name="BookService" …………..>
        <import
   namespace="urn:jwsnut.chapter2.bookservice/types/BookQuery"
                location="bookTypes.xsd"/>
        <import namespace="http://java.sun.com/jax-rpc-ri/internal"
                location="baseTypes.xsd"/>




        <message name="BookQuery_getAuthor">
                <part name="String1" type="xsd:string"/>
        </message>

…………………

</definitions>
```

<import> attributes

- The WSDL import element must have:

- `namespace` – the namespace which the definitions are to be imported into. This must match the target namespace defined in the imported schema

- `location` – a URI which indicates where the imported definitions can be found

Inline or imported?

- Imported types are not wrapped in the `<types>` element.
- It is possible to mix imported and inline definitions within the same document.
- Inline definitions are within `<types>` elements.

Mixed import, inline example

```
<import namespace="urn:jwsnut.chapter2.bookservice/types/BookQuery"
                      location="bookTypes.xsd"/>

<import namespace="http://java.sun.com/jax-rpc-ri/internal"
       location="baseTypes.xsd"/>

<types>
   <schema targetNamespace="…………………….">



   </schema>
</types>
```

Nesting inclusion of types

- There is also a XMLSchema import element which allows definitions to be referenced from one schema to another
- Similar to nested `#includes` in C++ header files
- This is different to the WSDL import element and inhabits the XMLSchema namespace
- The XMLSchema import element allows definitions from a different namespace to the target namespace for its parent schema

Schema import example

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace
   ="urn:jwsnut.chapter2.bookservice/types/BookQuery"...........>

       <import namespace="http://java.sun.com/jax-rpc-ri/internal"
              schemaLocation="baseTypes.xsd"/>


</schema>
```

Importing other types of definition

- The WSDL `import` element can be used to include all types of definitions that can appear in a WSDL document.

- Each set of definitions could be separated out into a different document. This can aid re-use.

- For instance the generic definitions of a web service can be separated from the `service` element.
  - This would allow a single service definition to describe several different instances of a service at different locations.