RDz Web Services Tutorial 01
Create a WSDL Description

Thanks to Mrs. Isabel Arnold, who conducted an IBM training session in Hamburg in August 2006, covering the material in this and the following 2 tutorials. The original Version of the material was created by Reginaldo W. Barosa, IBM Executive IT Specialist.

Content

1. Short Web Service overview

1.1 What is an XML Web Service?

The W3C defines a Web Service as a software system designed to support interoperable Machine to Machine interaction over a network. Web Services are frequently just Web APIs that can be accessed over a network, such as the Internet, and executed on a remote system hosting the requested services.

The W3C Web Service definition encompasses many different systems, but in common usage the term refers to

- **clients and servers that communicate using XML messages that follow the SOAP standard, and**
- **there is a machine readable description in the Web Services Description Language (WSDL) of the operations supported by the server.**

Some industry organizations, such as the Web Services Interoperability Organization (WS-I), mandate both SOAP and WSDL in their definition of a Web Service. Both are a prerequisite for automated client-side code generation in the mainstream Java and .NET SOAP frameworks.

XML Web Services offer the capability to describe interfaces in sufficient detail that users may create client applications that can communicate with the Web Services. These descriptions are contained in an XML-document named WSDL-document.

XML Web Services are registered. Potential users may find them using the UDDI-technology (Universal Description, Discovery and Integration) or an alternative directory service.

1.2 What is WSDL?

WSDL is an XML-based language for describing Web Services and how to access them. It is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate. Today, WSDL is used mostly in conjunction with SOAP.

WSDL is essentially an Interface Definition Language for SOAP.

The purpose of this tutorial is to illustrate the use of the WSDL editor to design a Web Services interface and produce new WSDL. There are many wizards in WDz that can be used to generate WSDL, but in this tutorial, we will start with an empty WSDL file and add appropriate information using the WSDL editor.

WSDL uses XML for describing Web Services. The description specifies

- **the characteristics of a Web Service:**
- **what the Web Service can do,**
- **where it resides, and**
- **how it is invoked.**

For an introduction into XML see
http://www.javaworld.com/javaworld/jw-04-1999/jw-04-xml.html?page=1

WSDL can be extended to allow descriptions of different bindings, regardless of what message formats or network protocols are used to communicate.

WSDL enables a service provider to specify the following characteristics of a Web Service:

- **Name of the Web Service and addressing information**
- **Protocol and encoding style to be used when accessing the public operations of the Web Service**
- **Type information: Operations, parameters, and data types comprising the interface of the Web Service, plus a name for this interface**

WSDL is a resumé for a business function implemented as a service. It describes what business functions are available, what information is to be pass to and from the business function, how the information is to get to the business function, and where the business function is located.

The business functions are represented by operations which are grouped into PortTypes. We can have as many operations as we would like in one PortType, but we would normally only specify closely related operations in a particular Port Type.

In this tutorial we will have a Port Type that represents functions a client can perform against data stored on a server (fund data). We are using a scenario of working for a mutual fund company, so the fund data is information our company has about the mutual funds it offers.

The operation (business function) we will add to our Port Type is addFund. (Optionally you can create additional WSDL files for getFund, deleteFund, and updateFund).

1.3 What is XSD

An XML Schema provides a means for defining the structure, content and semantics of XML documents. The XML Schema language is also referred to as XML Schema Definition (XSD).

We will use XSDs to describe the data layout of the information that is to be passed to and from the business operations. Our operations will receive and return messages. Messages may have one or more parts. A part may be a complex item with multiple elements. The message parts we will receive and send will point to the XSDs that describe the data. The XSDs contain multiple elements like Fund Id, Fund Name, Fund Price, etc.

The operations, messages, parts, and elements are used to generically describe our business functions, but do not tell us anything about their physical implementation. Binding information is added to the WSDL file to specify the protocol and transport used to access the operations. We are describing a Web Service. Web Services are accessed via SOAP over HTTP. WSDL can also be used to describe other types of 'services' that don't use SOAP over HTTP, but when discussing Web Services, you are discussing SOAP over HTTP.

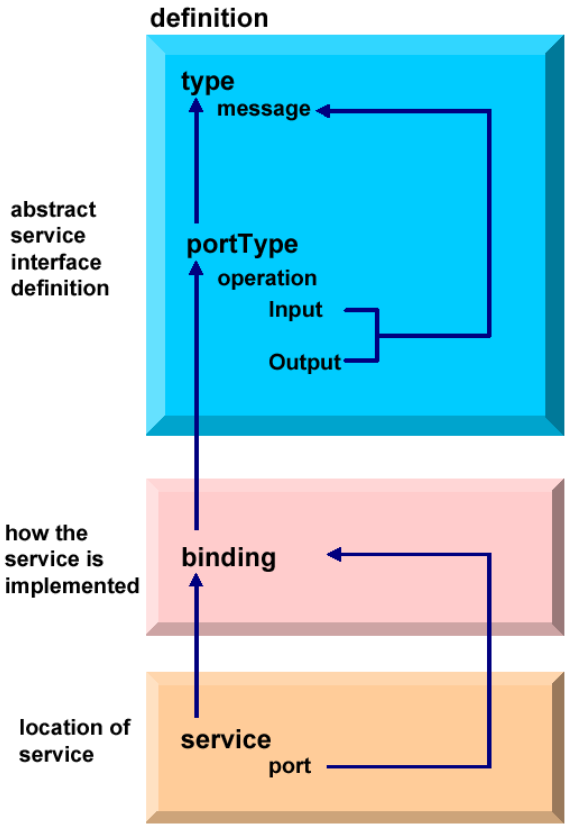
The last part of the WSDL file is the service information which details the location of the service.

1.4 WSDL Elements

A WSDL document contains the following six main elements:

1. **Port type. An abstract set of one or more operations supported by one or more ports. It includes an abstract description of an action supported by the service that defines the input and output message and optional fault message (section 2.6).**
2. **Message. An abstract, typed definition of the data being communicated. A message can have one or more typed parts (section 2.7).**
3. **Types. A container for data type definitions using some type system, usually XML Schema.**
4. **Binding. Specification of a concrete protocol and data format for a particular port type. The binding information contains the protocol name, the invocation style, a service ID, and the encoding for each operation Section 2.9).**
5. **Service. A collection of related ports. A WSDL document defines services as collections of network endpoints, or ports (section 2.10).**
6. **Port. A single endpoint, which is defined as an aggregation of a binding and a network address.**

WSDL supports the XML Schema Definition (XSD) specification.



WSDL elements and relationships

These are the elements comprising a WSDL document and the various relationships between them.

- **One WSDL document contains zero or more services. A service contains zero or more port definitions (service endpoints), and a port definition contains a specific protocol extension.**
- **The same WSDL document contains zero or more bindings. A binding is referenced by zero or more ports. The binding contains one protocol extension, where the style and transport are defined, and zero or more operations bindings. Each of these operation bindings is composed of one protocol extension, where the action and style are defined, and one to three messages bindings, where the encoding is defined.**
- **The same WSDL document contains zero or more port types. A port type is referenced by zero or more bindings. This port type contains zero or more operations, which are referenced by zero or more operations bindings.**
- **The same WSDL document contains zero or more messages. An operation usually points to an input and an output message, and optionally to some faults. A message is composed of zero or more parts.**
- **The same WSDL document contains zero or more types. A type can be referenced by zero or more parts.**
- **The same WSDL document points to zero or more XML Schemas. An XML Schema contains zero or more XSD types that define the different data types.**

1.4.1 Types

The types element encloses data type definitions used by the exchanged messages. WSDL uses XML Schema Definition (XSD) as its built-in type system:

```
<definitions ....>
   <types>
      <xsd:schema ..../>(0 or more)
   </types>
</definitions>
```

A simple types definition for example may have two schema sections: one defines the message format for the input and the other defines the message format for the output.

1.4.2 Messages

A message represents one interaction between a service requester and a service provider. If an operation is bidirectional, at least two message definitions are used in order to specify the transmissions to and from the service provider. A message consists of one or more logical parts.

```
<definitions ....>
   <message name="nmtoken">(0 or more)
      <part name="nmtoken"element="qname"(0 or 1)type="qname"(0 or 1)/>
                (0 or more)
   </message>
</definitions>
```

The abstract message definitions are used by the operation element. Multiple operations can refer to the same message definition.

1.4.3 Port types

A port type is a named set of abstract operations and the abstract messages involved:

```
<definitions ....>
   <portType name="nmtoken">
      <operation name="nmtoken"..../>(0 or more)
   </portType>
</definitions>
```

WSDL defines four types of operations that a port can support:

1. **One-way: The port receives a message. There is an input message only.**
2. **Request-response: The port receives a message and sends a correlated message. There is an input message followed by an output message.**
3. **Solicit-response: The port sends a message and receives a correlated message. There is an output message followed by an input message.**
4. **Notification: The port sends a message. There is an output message only. This type of operation could be used in a publish/subscribe scenario.**

1.4.4 Bindings

A binding contains:

- **Protocol-specific general binding data, such as the underlying transport protocol and the communication style for SOAP.**
- **Protocol extensions for operations and their messages include the URN and encoding information for SOAP, for example.**

A port references a binding. The port and binding are modelled as separate entities in order to support flexibility and location transparency. Two ports that merely differ in their network address can share the same protocol binding.

For a Web Service, the binding usually specifies SOAP. HTTP, MIME EJB, JMS, and plain Java transport bindings are other alternatives.

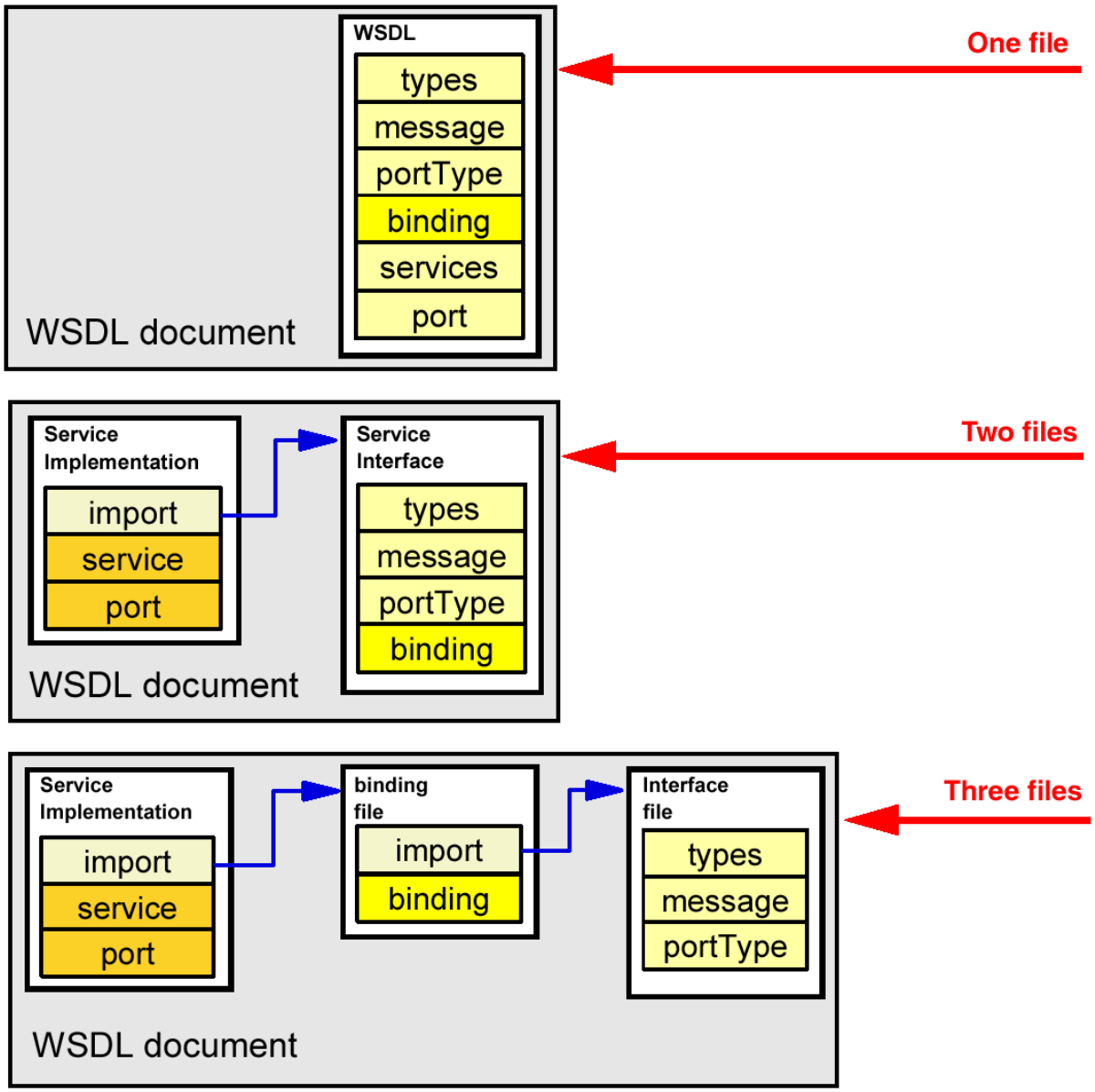
1.4.5 Port definition

A port definition describes an individual endpoint, usually an URI (for example an URL), by specifying a single address for a binding.

1.4.6 Service definition

A service definition merely bundles a set of ports together under a name,

1.5 WSDL document Overview



WSDL document Overview

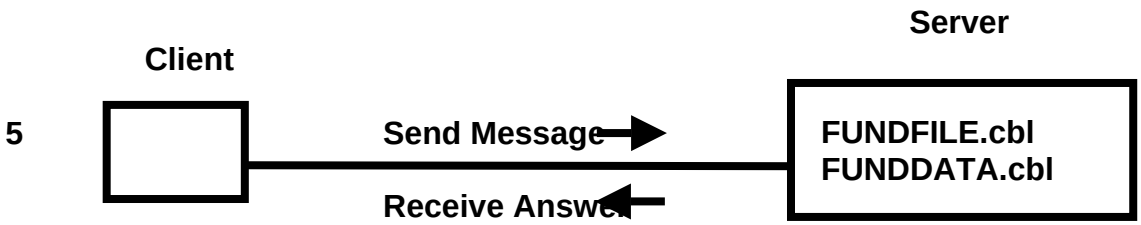
A WSDL document can be created in one or more physical files. If they are more than one, we have to connect these files using an import element. This separation of files can be convenient to hold the abstraction of concepts and to allow better maintenance.

Three files are typically used in WebSphere Developer for System z (and also in the underlying Application Developer).

2. WSDL-Editor Tutorial

Scenario

The purpose of this tutorial is to create a WSDL description, that can be used by a client to access a Webservice located on a server. In our scenario, you work for a mutual fund company. In order to make one of the existing business functions located on the server available as a Web Service, you will be describing the interface of the business function in WSDL.



Our Server contains a Web Service enabled COBOL package, including two datasets FUNDFILE.cbl and FUNDDATA.cbl. We will generate a WSDL description of a Web Service that permits the client by using this description to send a service request message to the server and receive an answer.

We will add an operation called addFund. The addFund operation will take an input and output message to be defined. The addFund operation will take all fields in the file as input, and will return the data that was added.

Tutorial Requirements:

Please note that there are often several ways to perform functions in WDz. This tutorial will present one of the ways. If you are familiar with WDz, you will notice that some of the statements are general, and not necessarily true for every situation. The main intent of this tutorial is to reinforce the discussions on SOAP and WSDL.

The following are other assumptions made in this tutorial.

Assumption 1: A VMWare image was supplied that contains WDz V7.0 and the files that are needed for this tutorial

Assumption 2: The VMWare image starts a Windows XP OS using “unilp” as login and password.

2.1 Tutorial Overview

1. Tutorial Overview (this section)

2. Initial Setup
In this part of the tutorial we will change the default line length in WDz for XML files. Although this step is not required, longer lines are required by the WSDL specification. The directions in part 1 have detailed information on why we want to extend the line length.

3. Create a Project
Our artifacts (various files and programs) are organized in WDz by projects. In this part of the tutorial we will create a simple project to hold the artifact we will use in this tutorial.

4. Create Data Schema Definitions
The information we will pass to and from our operations will ultimately go to a COBOL program. In this part of the tutorial we will generate some XSDs (XML Schema Definitions) that describe the layout of input and output of the COBOL program.

5. Create an Empty WSDL File
There are several wizards in WDz that could create WSDL for us, but to reinforce our discussions about WSDL and the parts of WSDL, we will start with an empty WSDL file and add all the information to the file that our application needs.

6. Add the addFund Operation
The business function that the WSDL will represent is to add a fund to a file. The name of this operation will be addFund.

7. Add Messages
The application data that is sent to and received from operations are called messages. Messages may have one or more parts. Each part may be a complex part with one or more elements. In this tutorial we will add messages that represent the data that will be passed to and from our addFund operation.

8. Add a fault to the addFund Operation
The usual way to indicate a problem with an operation is to return a fault. In this part of the tutorial we will add information about the fault that will be returned if a problem occurs with the addFund operation.

9. Add Binding Information to the WSDL File
After we have indicated our business operation and the information that goes to and from our operations, we need to specify how to access our business operation. WSDL Binding information is used to indicate the protocol and transport that should be used. In this tutorial we construct WSDL for a Web Service. A Web Service is SOAP over HTTP. There are also other types of services that use other protocols and transports.

10. Add Service Information
We also need to specify where the Web Service is located. The location of the Web Service is specified in the Service information.

11. (Optional) Create more WSDL files
As an optional exercise, create a WSDL file (one each) for the getFund, updateFund, and deleteFund services.

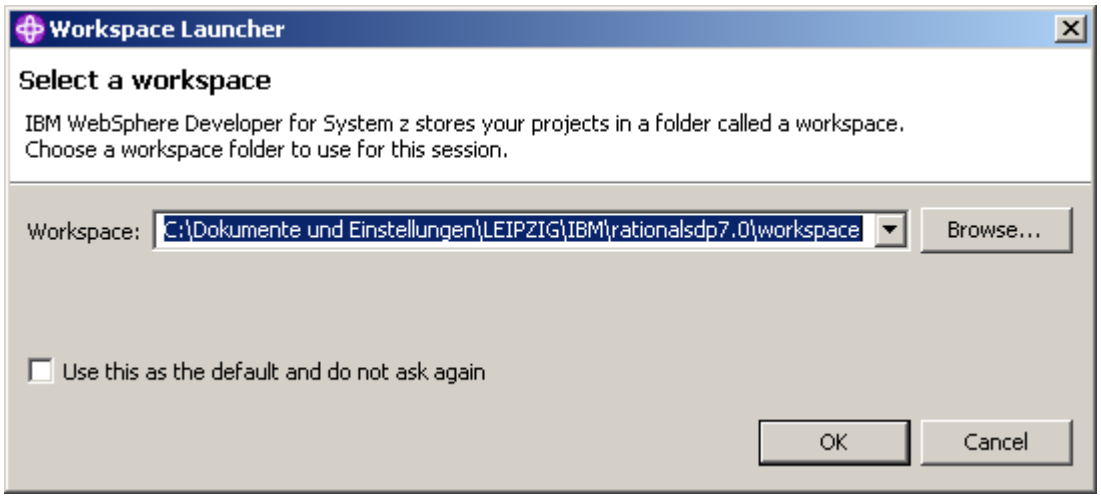
2.2 Initial Setup

The default width for the XML editor is 72 characters. Some of the lines we will be generating need to be longer than 72 characters to be compliant with the WSDL specification. To avoid syntax problems we will change the XML editor width to 100 characters. This is a one-time setup for the workspace.

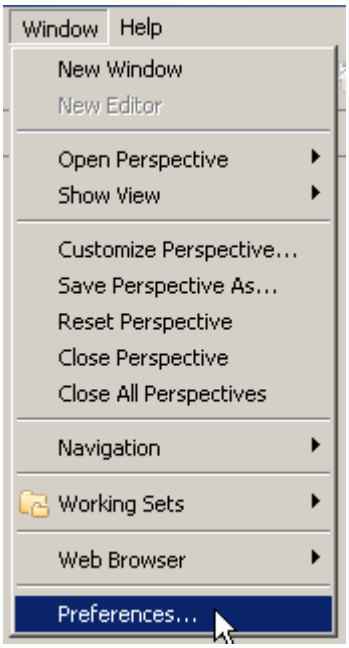
Note that if we do not change this, the wizard will still properly add code, but some lines may be broken into two or more lines where the WSDL specification does not allow lines to be broken. Although we could manually go in and correct the problem (or use short names), it is easier to change the XML editor line width.

Note also that WSDL is normally stored on a PC or in USS files on z/OS, so expanding the character width doesn't present any problems.

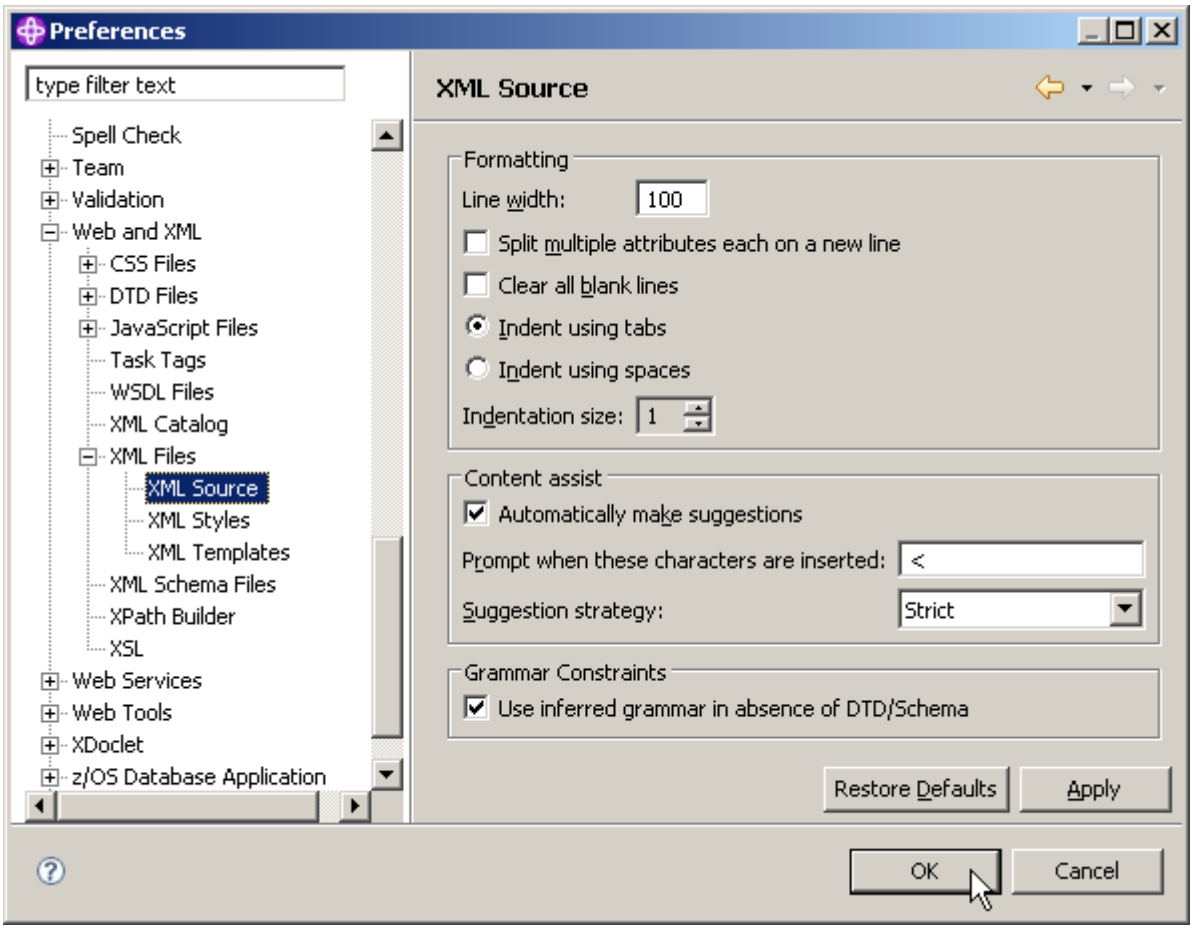
1.Start WDz if it is not already running.



2.If asked, Use the default workspace as proposed. Click OK.



3.From the menu bar, select Window → Preferences...

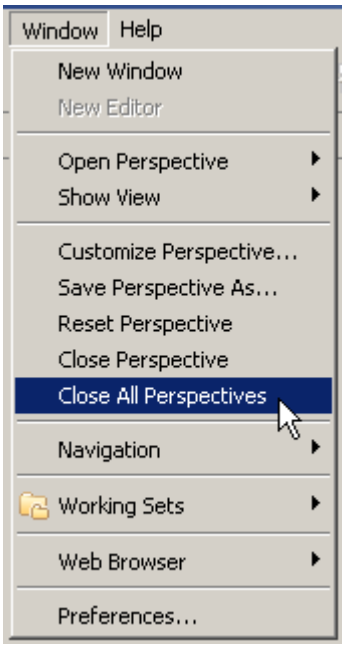


4.In the navigation on the left, Select Web and XML → XML Files → XML Source.
On the right, change the Line width to 100, then click OK.

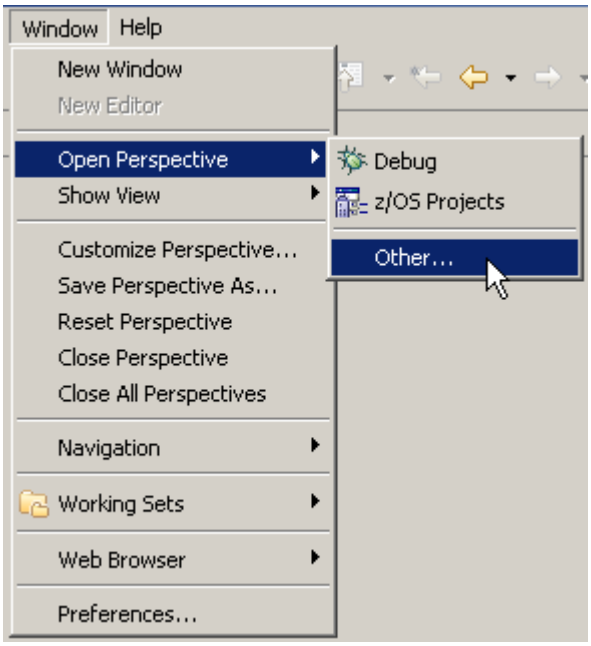
2.3 Create a Project

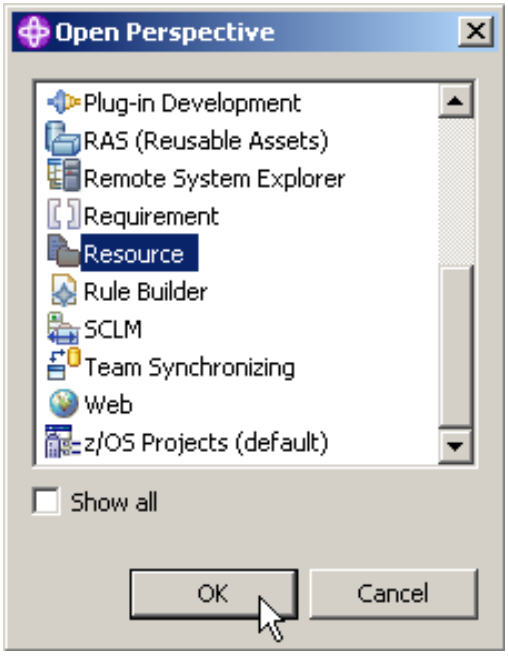
Computer artifacts are physical files that execute or are used by your software. Artifacts may include specifications, designs, code, use cases, class diagrams, etc. An artifact is one of many kinds of tangible byproduct produced during the development of software.

The various artifacts that we create in WDz are organized into projects. In this part of the exercise, we will create a simple project named WSDLCreate. We will be working from the Resource Perspective, but the functionality we will be using is available in any type of project in any type of perspective. The menus and wizards we will be using look at file extensions to determine the functionality available for this file.



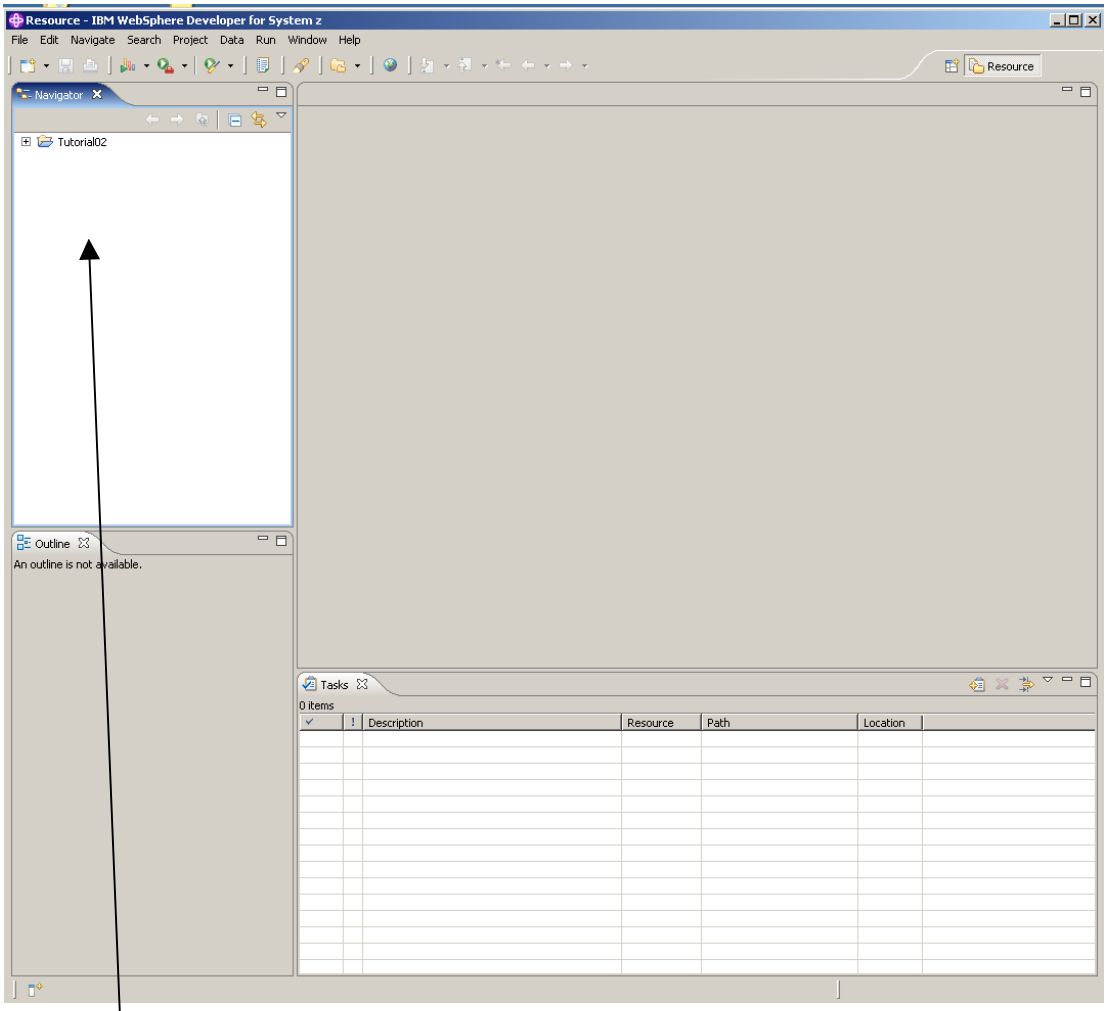
5.From the menu bar, select Window → Close All Perspectives.



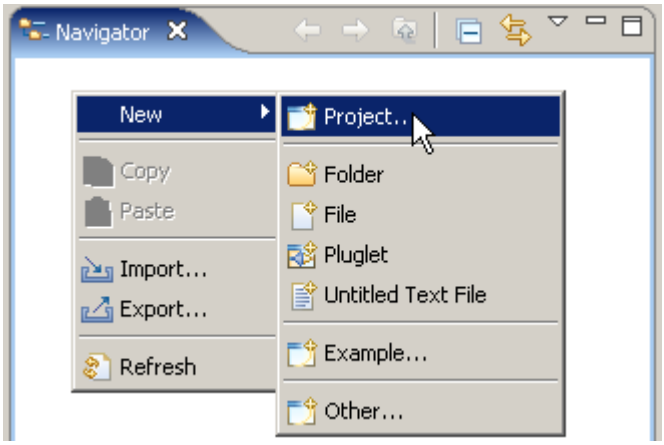


6.Then select Window → Open perspective → Other... and select Resource to open the resource perspective.

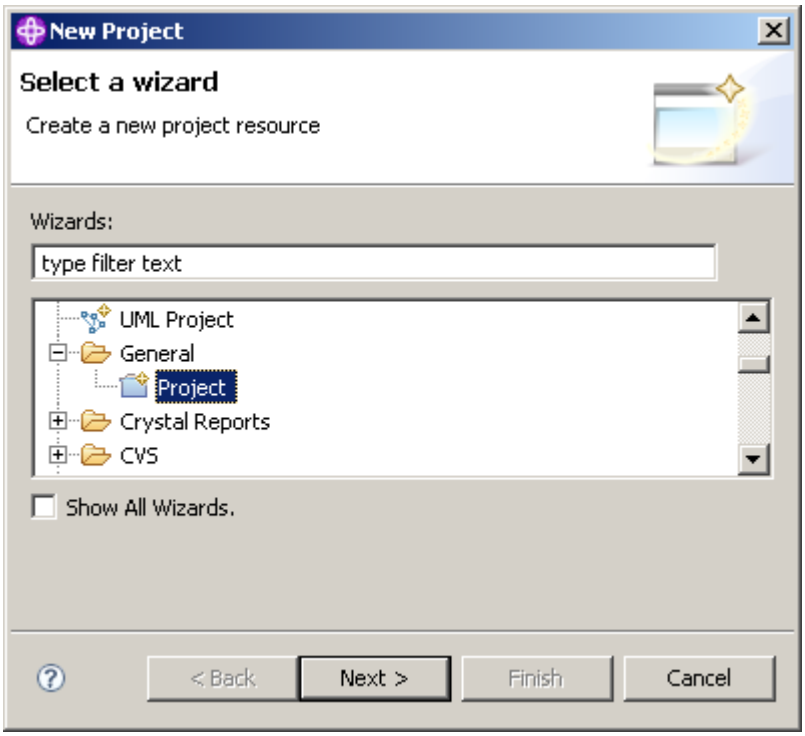
Click OK.



The screen now looks like shown above.:

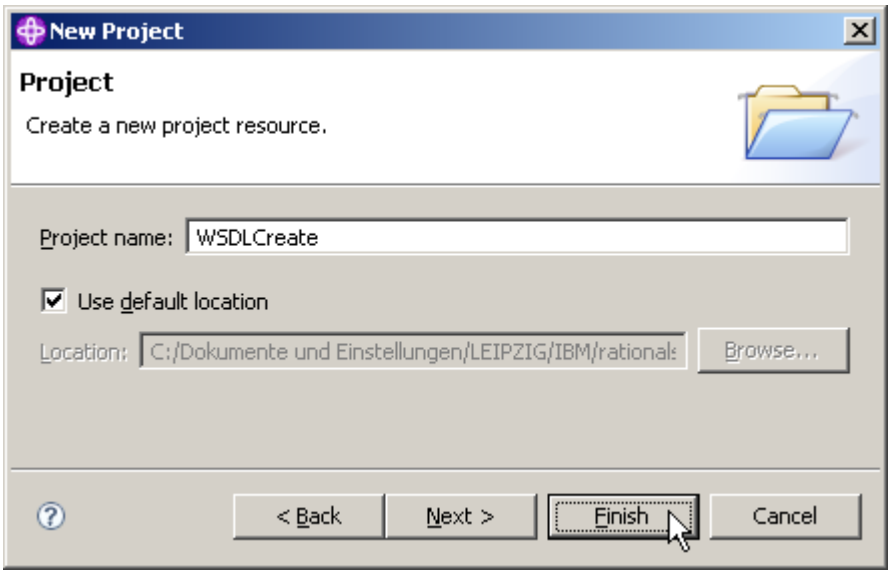


7. In the Navigator view, right-click in an open area, and from the context menu select New → Project…

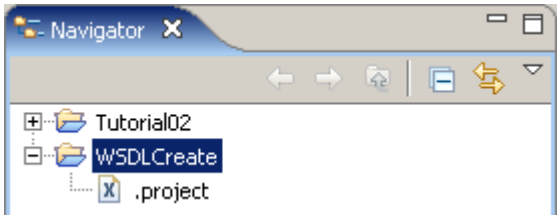


8.Expand General and select Project.

Click the Next button.



9.We decide to name our new Project WSDLCreate. On the New Project panel, specify the Project name as WSDLCreate, ensure Use default location is checked, then click the Finish button.



The Navigator view now contains a new entry WSDLCreate.
And if you expand it you will see that it is empty besides the project description file .project.

2.4 Create Data Schema Definitions

We will first import the COBOL program FUNDPROG.cbl and copybook FUNDFILE.cbl.

FUNDPROG.cbl is a regular Cobol program that is supplied with this tutorial. You may want to use a text editor to have a look at both FUNDPROG.cbl and FUNDFILE.cbl .

FUNDFILE.cbl is a Cobol copybook A copybook can be a copy of a part of any source code: working-storage, environment division, procedure division, etc. If there is source code in a COBOL program, which will be used in multiple programs, it is often put in a copybook.

A copybook is most frequently used to define the physical layout of data (either in an input or output file (file schemas), or in a temporary area (working storage)), but can also be used for sections of procedural code.

Data files, used by COBOL programs, are formatted according to record structures that can be defined in COBOL copybooks. A COBOL copybook declares the name and data types of variables that associated COBOL programs use to exchange information via the program's I/O Area (IMS) or COMMAREA (CICS). A copybook may be included in more than one COBOL program and a COBOL program can import more than one copybook. External applications passing information to the COBOL programs have to use a format declared in the imported copybook definitions.

A COBOL Copybook describes that data. The reason for using it is to allow multiple programs to reference the same structure without recoding in each program. It works like an INCLUDE statement, and it is up to the programmer to make sure the data complies with the description.

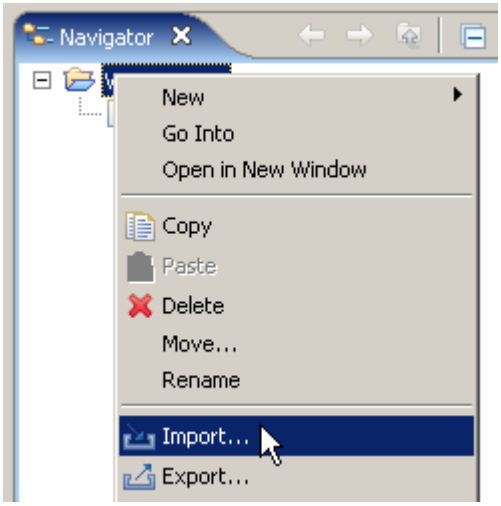
The information that will be passed to and from our Web Service will be similar to the data that is used in our target COBOL program. In this part of the exercise we will create some XSDs (XML Schema Definitions) that correspond to the Data Definitions in our COBOL program.

XML-Schema (bzw. XSD = XML-Schema-Definition) ist die moderne Möglichkeit, die Struktur von XML-Dokumenten zu beschreiben. XML-Schema bietet auch die Möglichkeit, den Inhalt von Elementen und Attributen zu beschränken, z. B. auf Zahlen, Datumsangaben oder Texte, z. B. mittels regulärer Ausdrücke. Ein Schema ist selbst ein XML-Dokument.

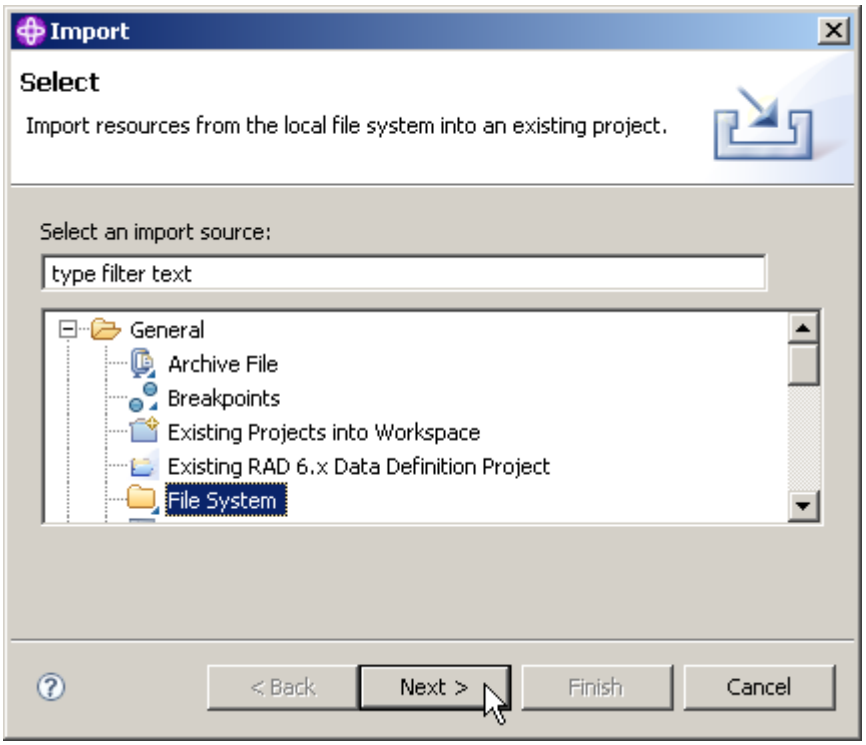
(In a later part of this tutorial, we will use the XSDs as descriptions of the messages that are passed to and from the Web Service operation.)

The WSDL editor in WDz allows you to define complex data layouts in XSD from scratch, but in this tutorial we will use an easier approach, and will generate corresponding XSDs from the existing COBOL data definitions.

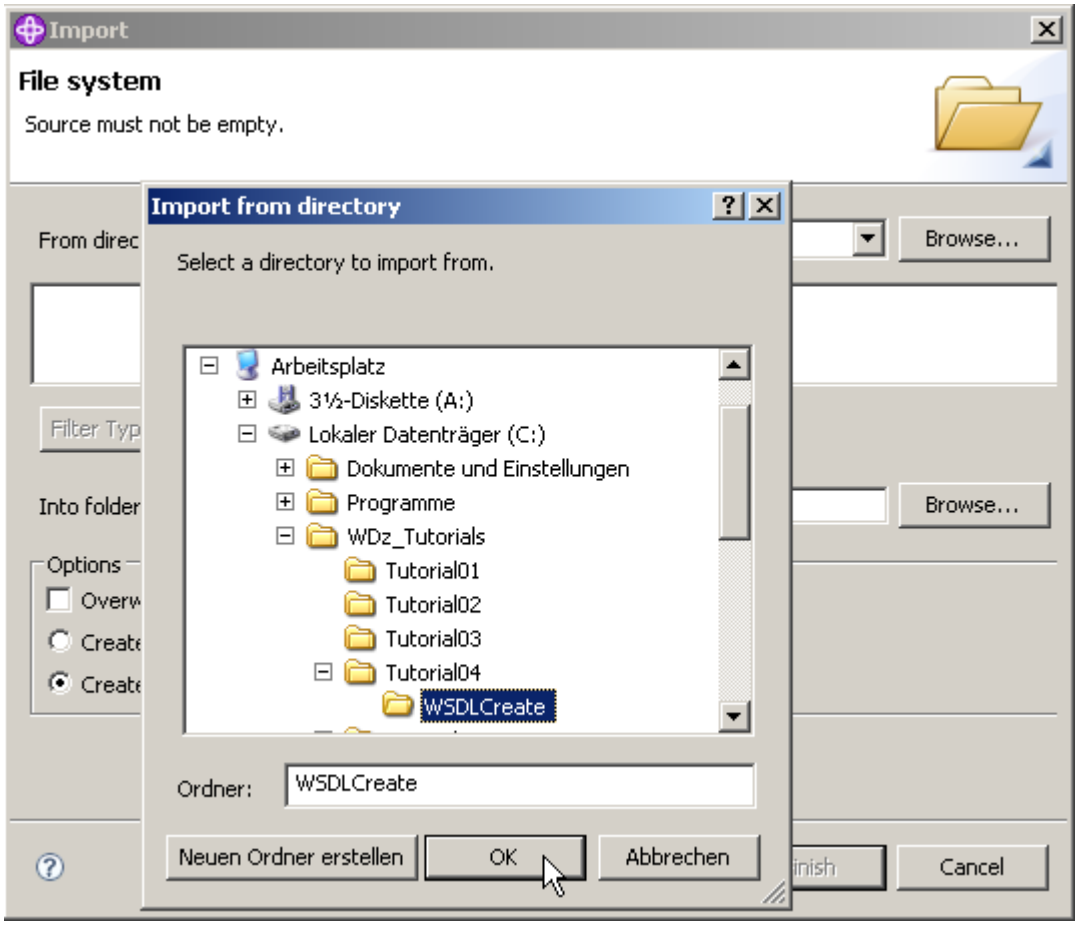
We will then use a wizard to generate the XSDs from FUNDPROG.cbl and FUNDFILE.cbl. This will save us the effort to generate the WSDL manually. However, the intent of this tutorial is to reinforce concepts discussed in the XML, SOAP, and WSDL lectures.



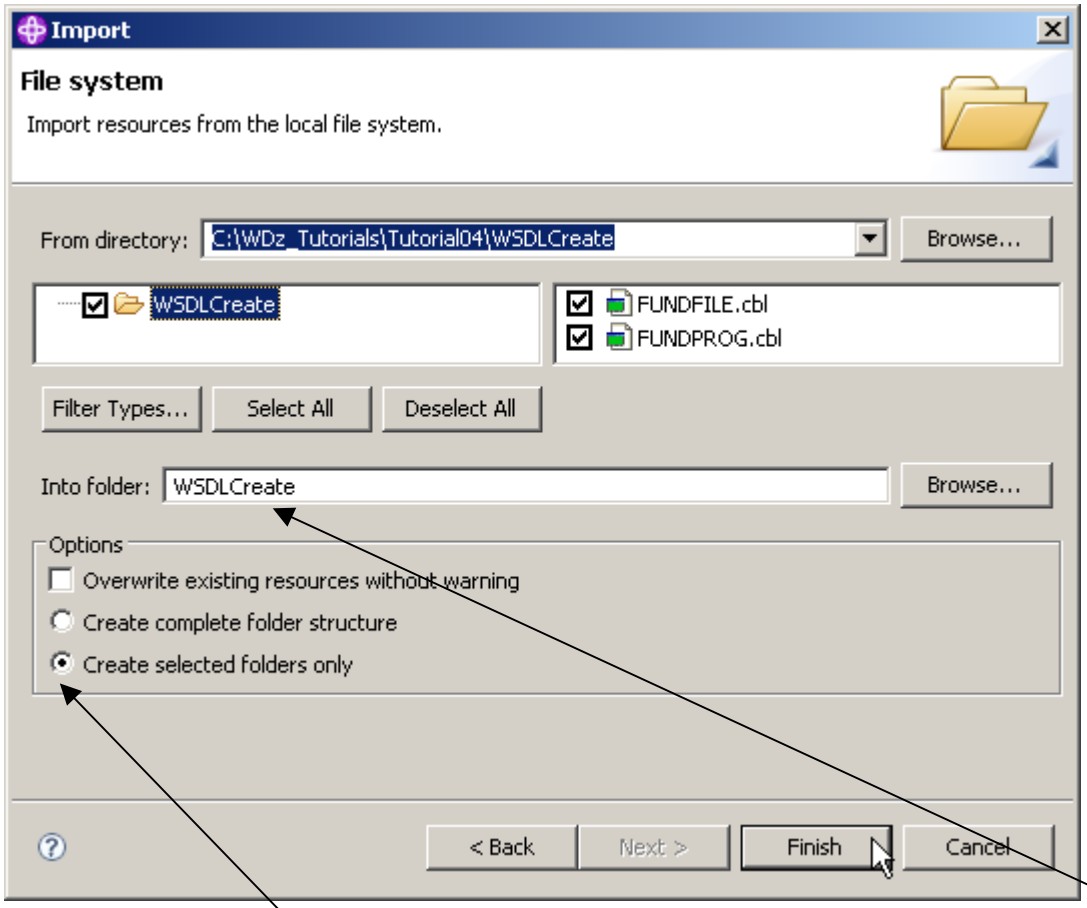
10.From the Navigator view, right-click the WSDLCreate project, and from the context menu, select Import...



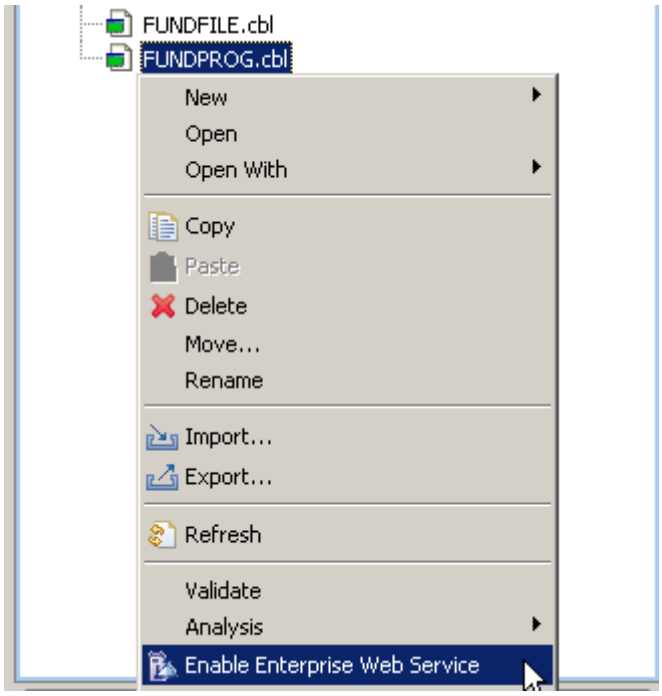
11.On the Select panel of the Import wizard, expand General and select File System. Click the Next button.



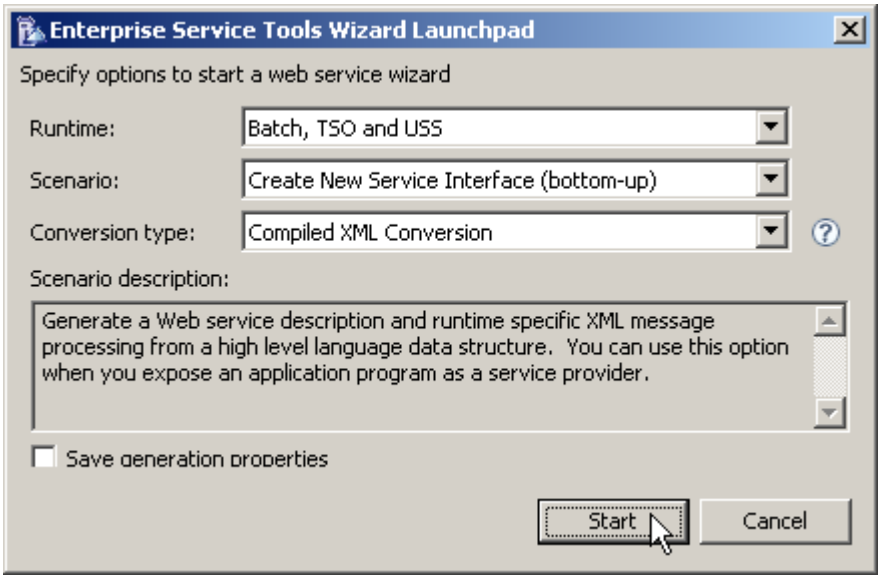
12.Click the Browse button on the top right and navigate to C:\WDz_Tutorials\Tutorial04\WSDLCreate and click OK.



13.Back on the File system page of the Import wizard, check WSDLCreate, ensure that the Into folder is WSDLCreate, ensure that Create selected folders only is selected and click the Finish button.

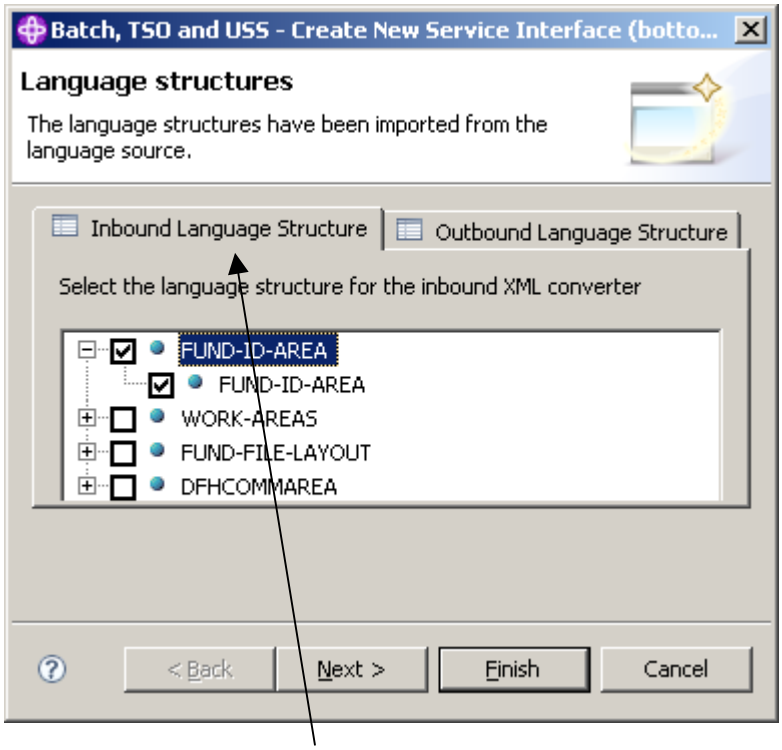


14.From the Navigator view, expand WSDLCreate, right-click on FUNDPROG.cbl and select Enable Enterprise Web Service.



15.On the Enterprise Service Tools Wizard Launchpad panel, set Batch, TSO and USS as Runtime, Create New Service Interface as Scenario and Compiled XML Conversion as Conversion Type.

Click Start.



16.On the Language structures panel, select the Inbound language structure tab, and check FUND-ID-AREA (if not already checked by default).

Do not yet klick on Next or Finished.

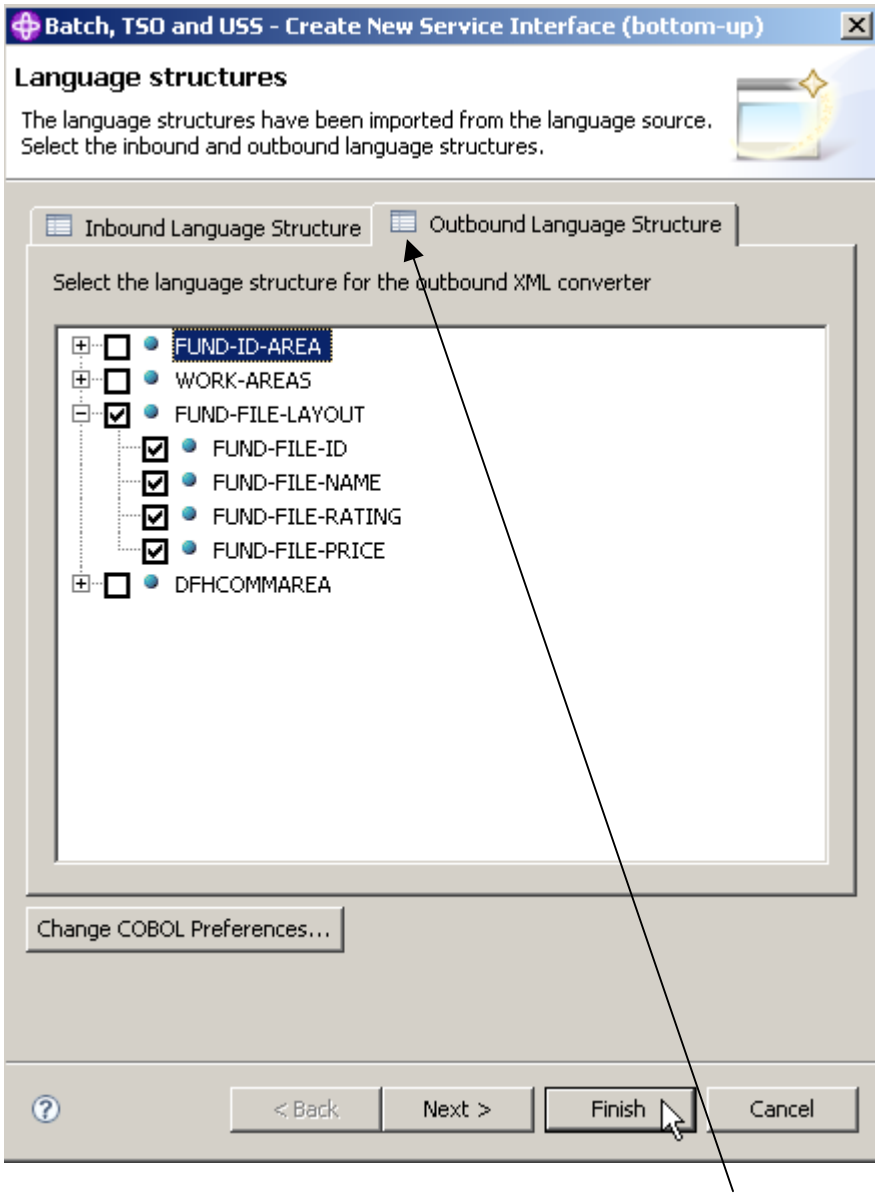
Note: The Cobol Program FUNDPROG.cbl contains as usual a Data Division. The code in the Data Division is shown on the next page.

FUND-ID-Area is the parameter in the WORKING-STORAGE-SECTION of the DATA-DIVISION of your Cobol Program FUNDPROG.cbl.

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  FUND-ID-AREA                    PIC X(8).
01  WORK-AREAS.
    05  RESP-FLD                    PIC S9(8) COMP.
    05  RESP-FLD-TWO                PIC S9(8) COMP.
    05  APP-ID                      PIC X(10) VALUE 'FUNDPROG:'.
    05  CSMT-MSG                    PIC X(121).
    05  DISPLAY-NUMBER              PIC 9999.

COPY FUNDFILE.
```

Note: Although we are going through a wizard that could generate WSDL, we are only using the wizard to generate XSDs (XML Schema Definitions) to describe a couple of data definitions in the COBOL program. FUND-ID-AREA describes fund id, which is the key to the fund file. This key would be the input message for a delete WSDL, or read WSDL operation. To use this as a message in WSDL, it must first be described in an XSD.

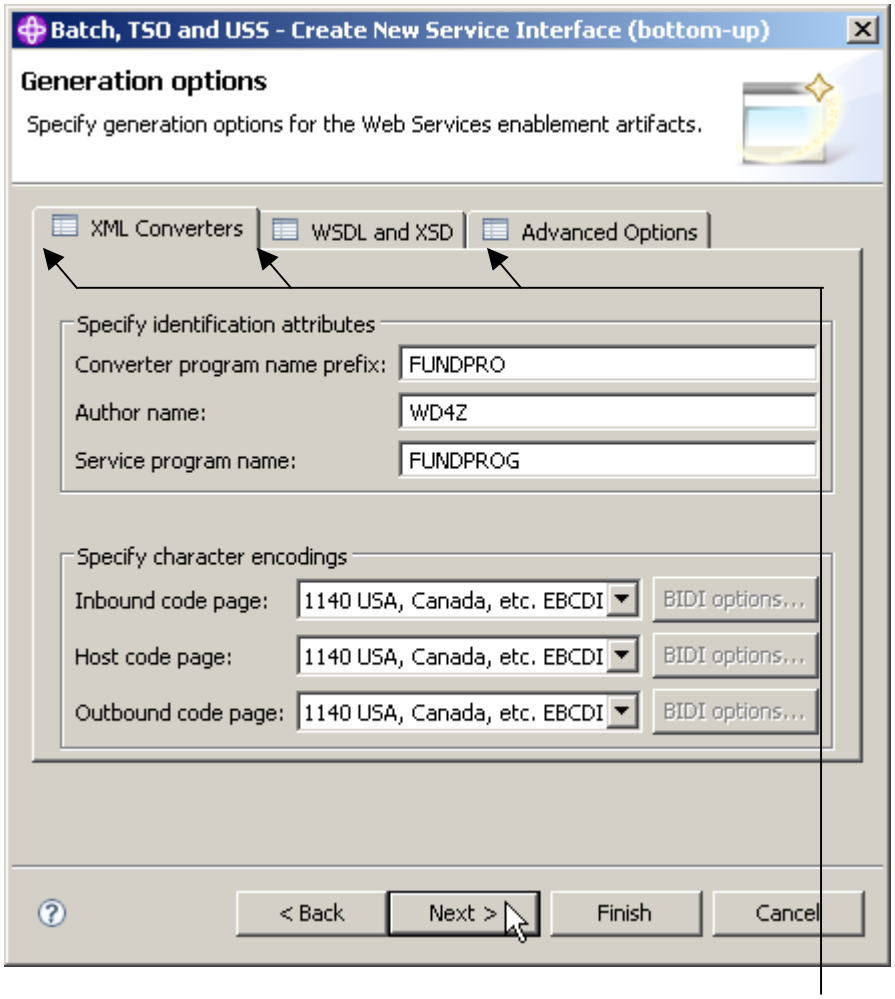


17.Still on the Language structures panel, switch to the Outbound language structure tab and check FUND-FILE-LAYOUT.

FUND-FILE-LAYOUT is the layout of the output message of our Web Service. FUNDFILE.cbl contains the following information:

```
01  FUND-FILE-LAYOUT.
    03  FUND-FILE-ID                PIC X(8).
    03  FUND-FILE-NAME              PIC X(50).
    03  FUND-FILE-RATING            PIC X.
    03  FUND-FILE-PRICE             PIC X(15).
```

Click Next (not Finish)



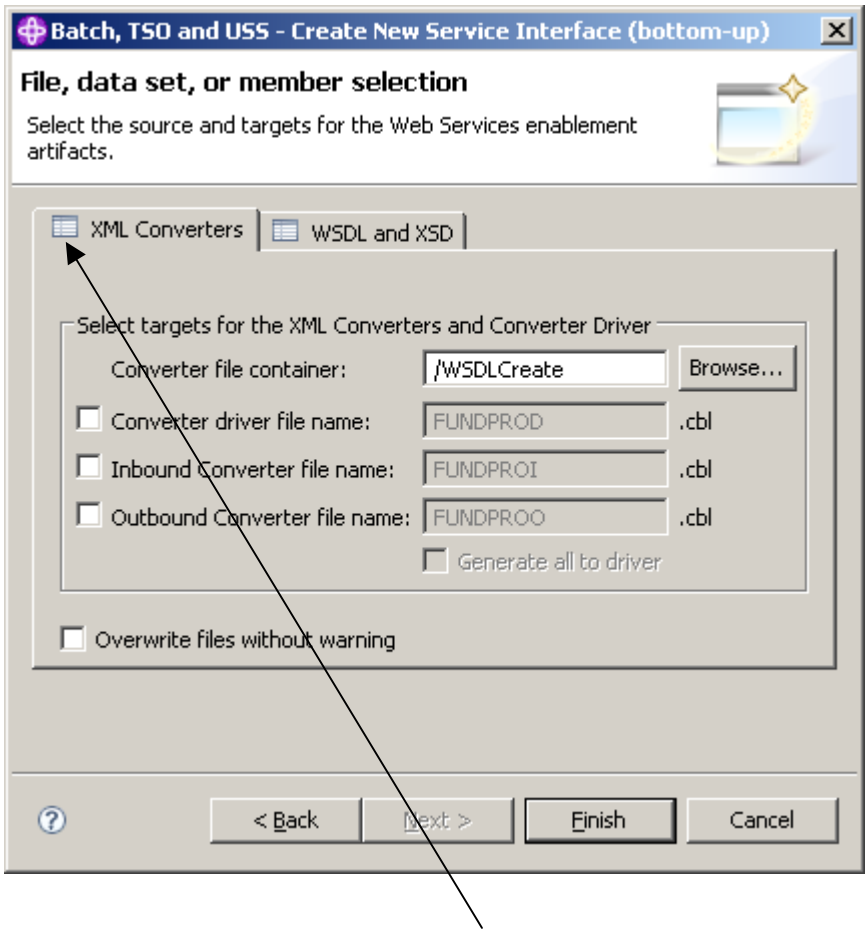
18.On the Generation options panel, feel free to have a look at the various options of the three tabs:

- **XML Converters**
- **WSDL and XSD**
- **Advanced Options**

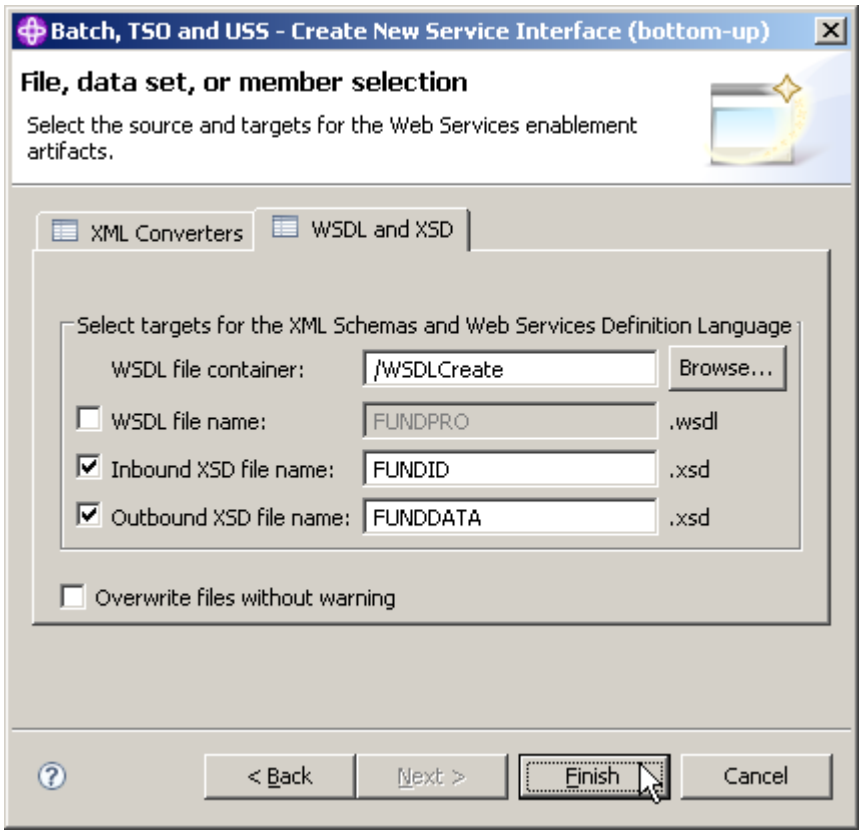
When you're finished, click Next.

Note: We will ask the wizard to generate XSDs for us, so we won't be generating a Converter program. (We will be generating a converter program in a later tutorial). From the WSDL and XSD Options tab, if we were generating WSDL, we could set the endpoint here. We could also set the Inbound and Outbound namespace names for the XSDs we will generate, but for this tutorial, the defaults are OK.

Klick next



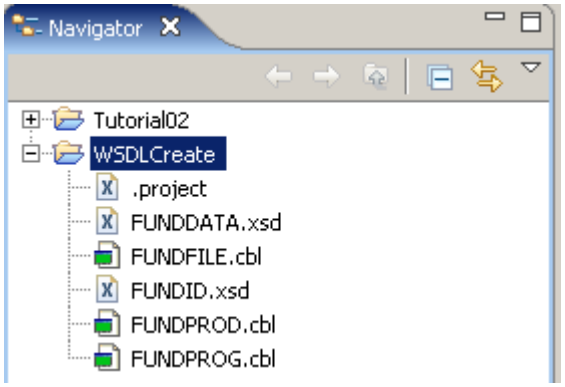
19.On the File, dataset, or member selection panel, click the XML Converters tab and uncheck all checkboxes. We are unchecking these options because we only want the wizard to generate XSDs for us.



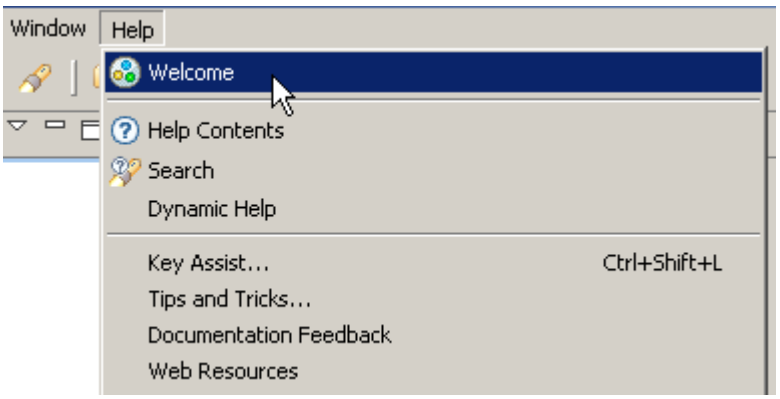
20.Still on the File, dataset, or member selection panel, switch to the WSDL and XSD tab and uncheck the box next to WSDL file name. Change

- **Inbound XSD file name to FUNDID and**
- **Outbound XSD file name FUNDDATA.**

Click the Finish button.

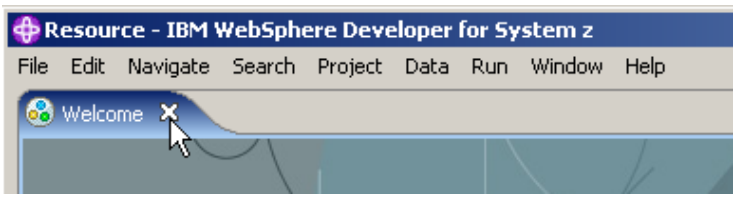


21.You should see a FUNDID.xsd and FUNDDATA.xsd file in your WSDLCreate project. Feel free to open each of the files and have a look at the content.



22.At this point, you should verify if the neccessary User Roles are enabled.
In the menu bar, click on Help ? Welcome.

23.Click on the Symbol vaguely looking like a human body in the right lower corner.
The Icon for Web Developer (advanced) must be activated. If not, activate it by clicking on it.



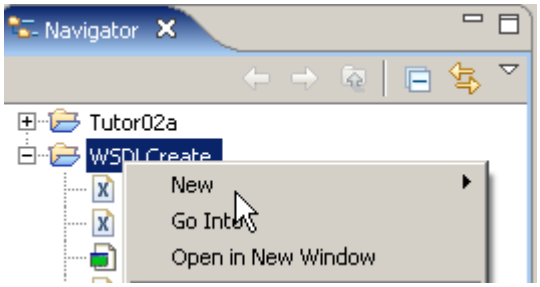
Close the Welcome window.

2.5 Create an Empty WSDL File

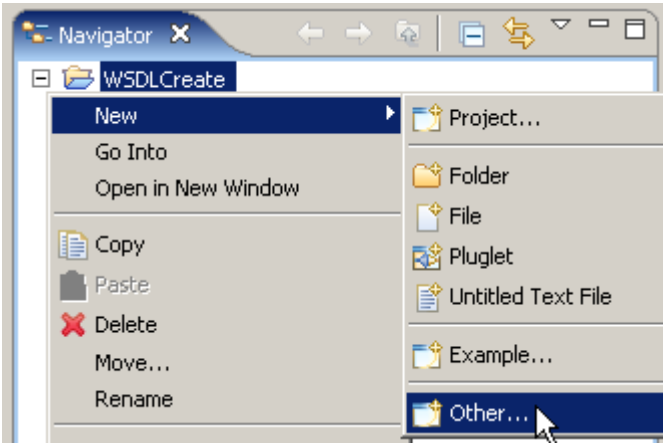
In the previous part of this tutorial, we generated some XSDs. One XSD represents the Fund file ID, and the other XSD represents the layout of the Fund file (all fields). We will use these XSDs as messages in the WSDL file that we will create.

In this part of the tutorial we will create an 'empty' WSDL file. Initially, the WSDL file will not have any operations, messages, bindings, or services. We will add those parts in a later part of this tutorial. This initial WSDL file will contain basic namespaces.

Namespaces are external locations that contain definitions for various parts of the WSDL we are defining – including the elements, attributes, and types. These definitions define various parameters including SOAP (protocol) and Envelop (data or payload).

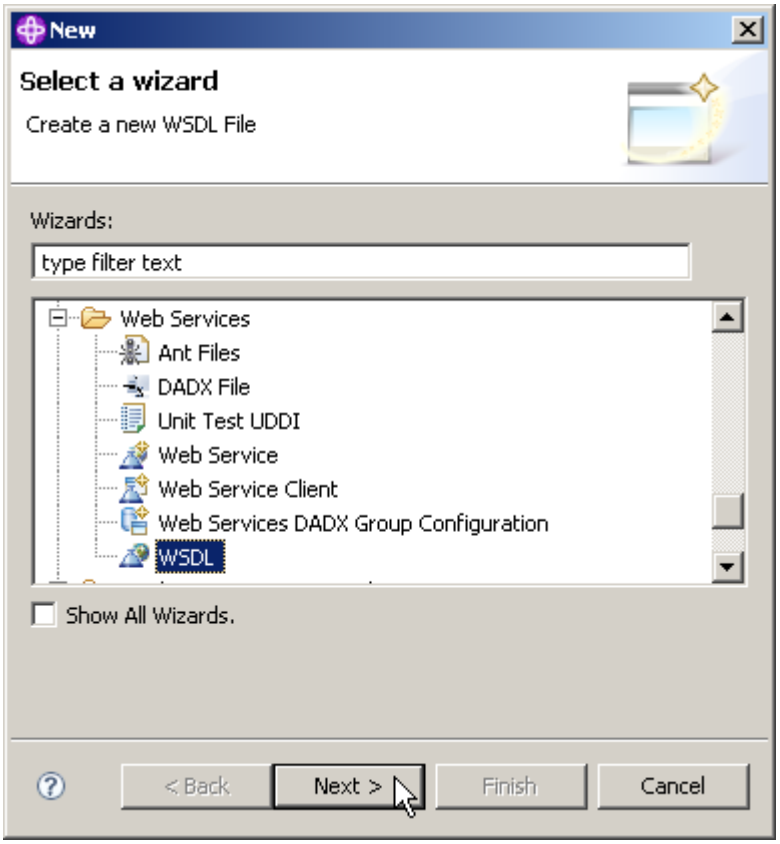


24.Right-click your WSDLCreate project.
From the context menu select New …

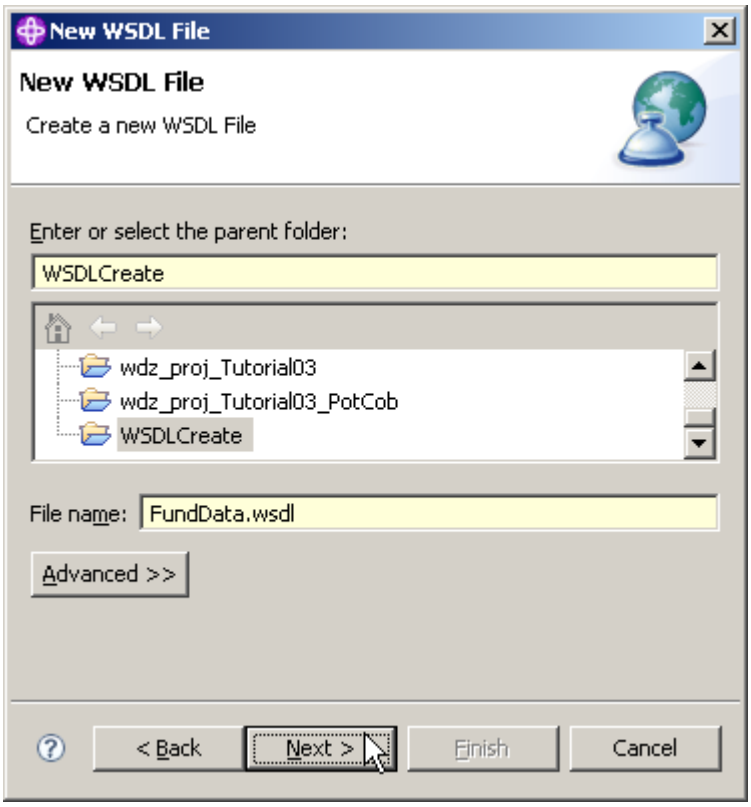


and Other...

scroll down

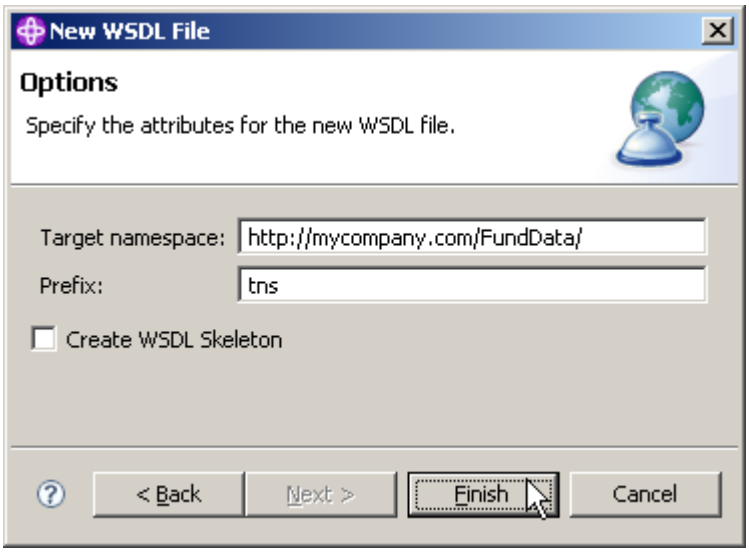


25.On the Select a wizard panel, select Web Services ? WSDL, then click the Next button.



26.On the New WSDL File panel, ensure that the parent folder is WSDLCreate, specify the File name as FundData.wsdl, and click Next.

Note: if you performed earlier WDz Tutorials, your screen may differ from the one above.



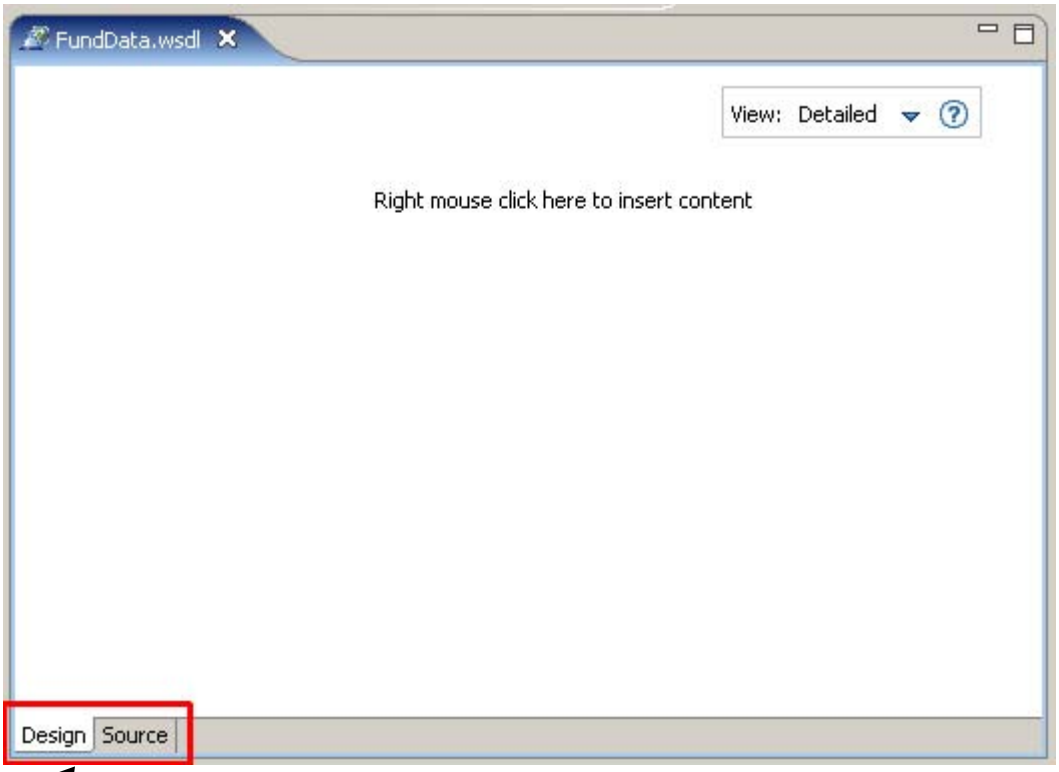
27.On the Options dialog, specify

- **http://mycompany.com/FundData/ for Target namespace,**
- **tns for Prefix,**
- **uncheck Create WSDL skeleton**

and click the Finish button.

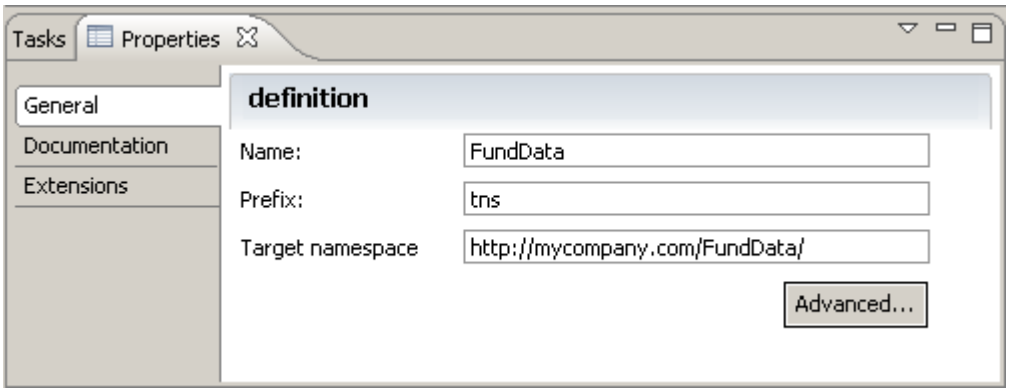
Note: We unchecked the Create WSDL Skeleton option because we want to add the operations, messages, bindings, etc, manually.

Click Finish



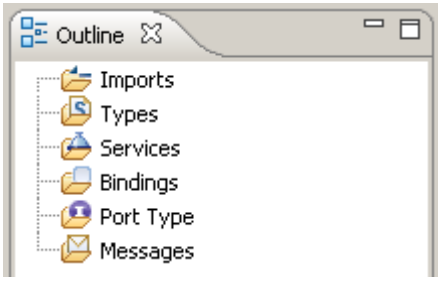
28.The FundData.wsdl file is now added in the Navigator view and opened with the WSDL-Editor

Note: This ist the WSDL-Editor window. On the bottom of the editor are two tabs. You are currently looking at the Design view, which shows a graphical representation of the WSDL file. There is also a Source view where you can view and edit the text that makes up the WSDL file.



29.It is important to note that the Properties view shown above is closely associated with the WSDL-Editor. The characteristics of items you click on in the WSDL editor will be displayed in the Properties view.

Note: If the properties view is not enabled, just open it from the menu bar (Window → Show view → Properties).



30.This is the Default Outline view situated below the Navigator view. You can use it to navigate through the different WSDL elements. It is also very useful.

2.6 Add the addFund Operation

In this part of the tutorial we will generate a PortType. We will add an operation called addFund.

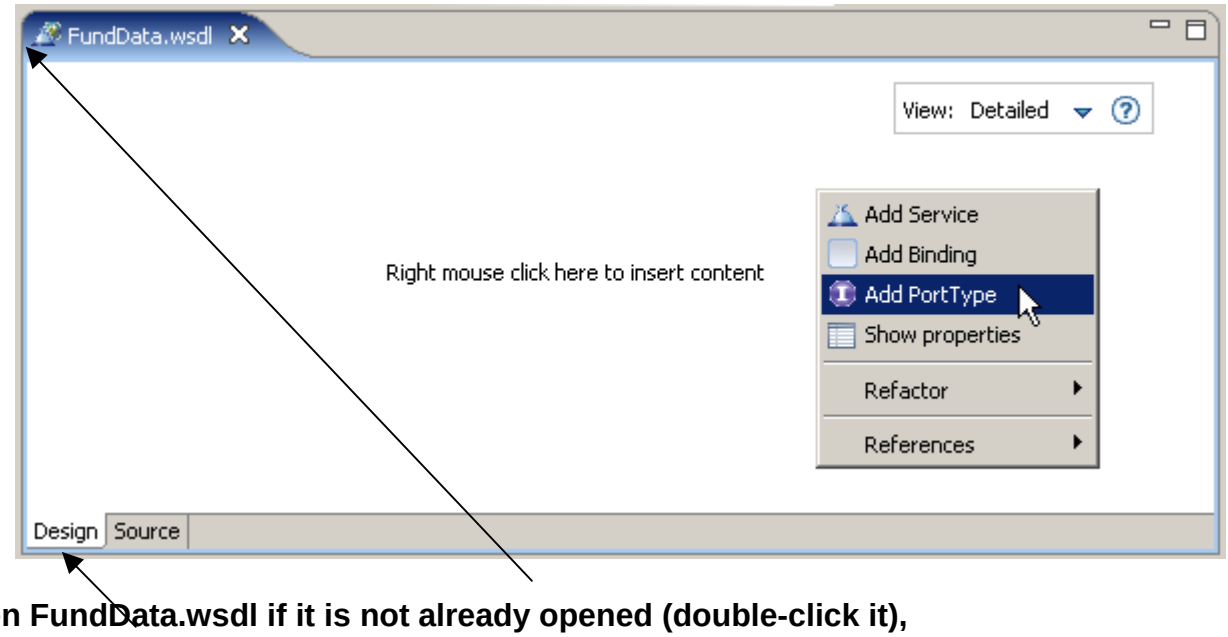
A PortType is sometimes called an "Interface". Actually, there are some similiarities with the function of an interface in RMI, Corba, or DCE. On the other hand, WSDL is also an interface, albeit much richer than an RMI or Corba Interface.

The addFund operation will take an input and output message that we will define later. The messages will point to the XSDs we created in the first part of this tutorial. The addFund operation will take all fields in the file as input, and will return the data that was added. Other design alternatives might be to return nothing, or return the ID of the Fund that was added.

It is common to have an add service return what it added. Depending on how you combine your Web Services, you will normally find that you want the output of one service to become input to the next service. By returning the added values from the add service, you can have those values flow into the next step of your business process.

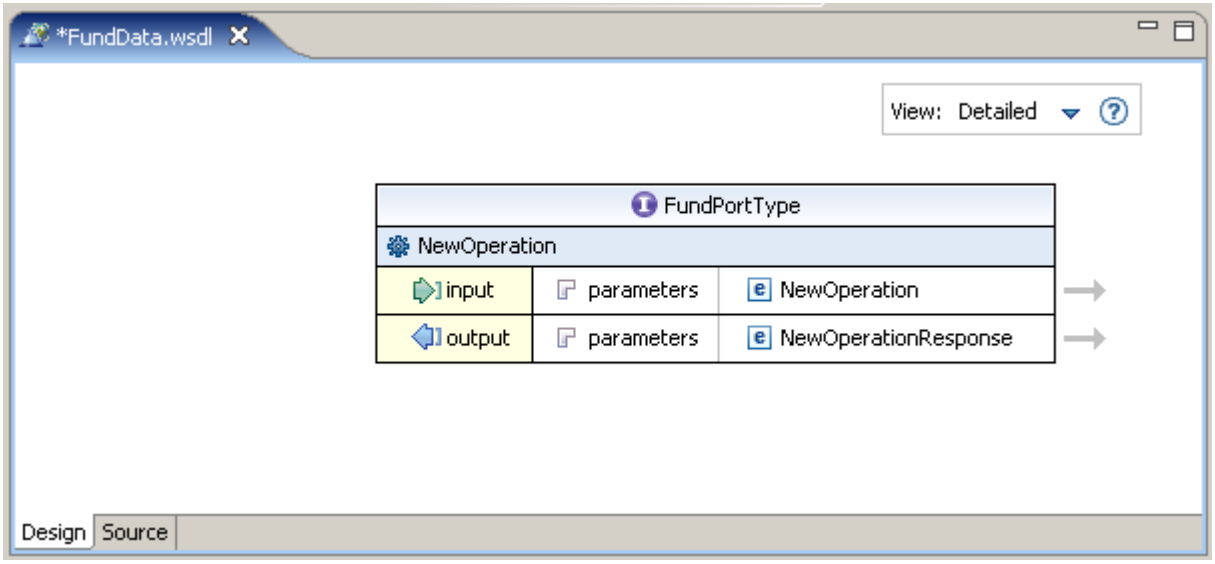
Our addFund operation is also capable of returning a fault if the operation is unsuccessful. The fault data is a string that indicates the reason why the fund could not be added.

We will first add a Port Type to the Port Types area. We will then add the addFund operation to the Port Type. The addFund operation will have input, output, and a fault. We will then associate the input, output, and fault with Messages.



31.Open FundData.wsdl if it is not already opened (double-click it), and switch to the Design view.

32.Right-click into the empty file and from the context menu, select Add PortType.

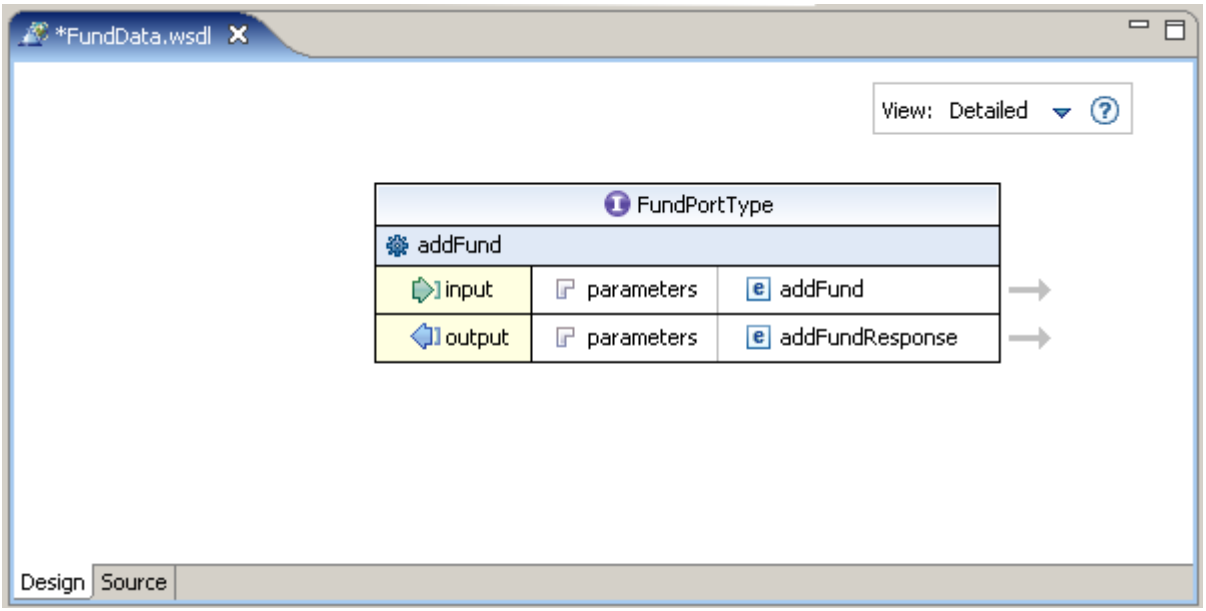


33.A new element appears, the PortType. Change the name from NewPortType to FundPortType.

Note: There are several ways to change the name of your freshly added port type. You can just click into the name to make it editable, or you can make the changes in the above mentioned Properties view, just to name two possibilities.

Klick auf NewOperation und dann auf den Pfeil, dann kann man den Namen im Properties Window ändern. direcktes Reinklicken funktioniert nicht.

As you can see, the added PortType has a NewOperation including input and output indicators.

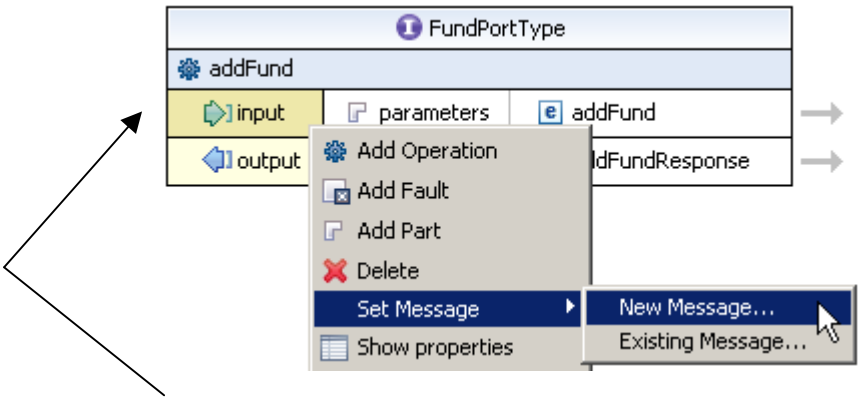


34.Rename NewOperation to addFund the same way you did with the PortType.

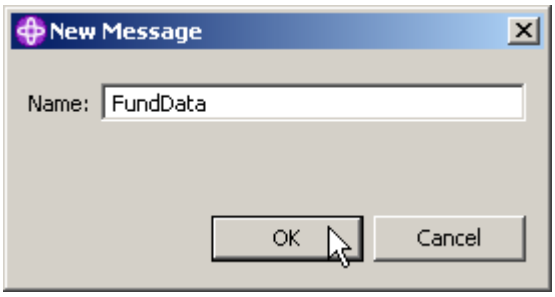
35.Save the changes with CTRL + S.

2.7 Add Messages

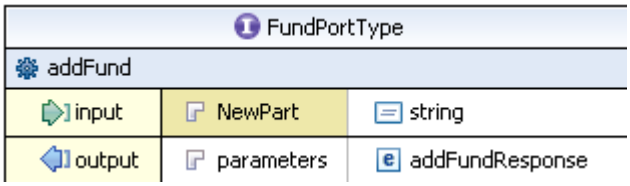
In this part of the tutorial, you will add messages to your WSDL. Messages are the application data that is passed to and from operations. The input and the output of our addFund operation will be all the fields in the Fund file. You created the message layout (XSDs) in chapter 2.4 of this exercise. The XSD that describes the data in the Fund file is in a file called FUNDDATA.xsd.



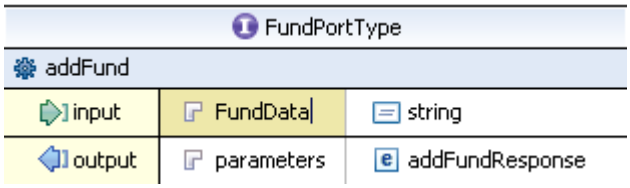
36.To add a message, right-click the input element and select Set Message → New Message...



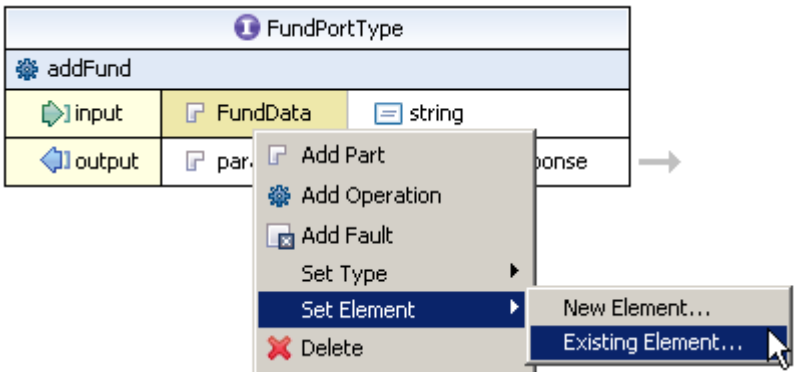
37.In the New Message dialog, specify the Name as FundData and click the OK button.



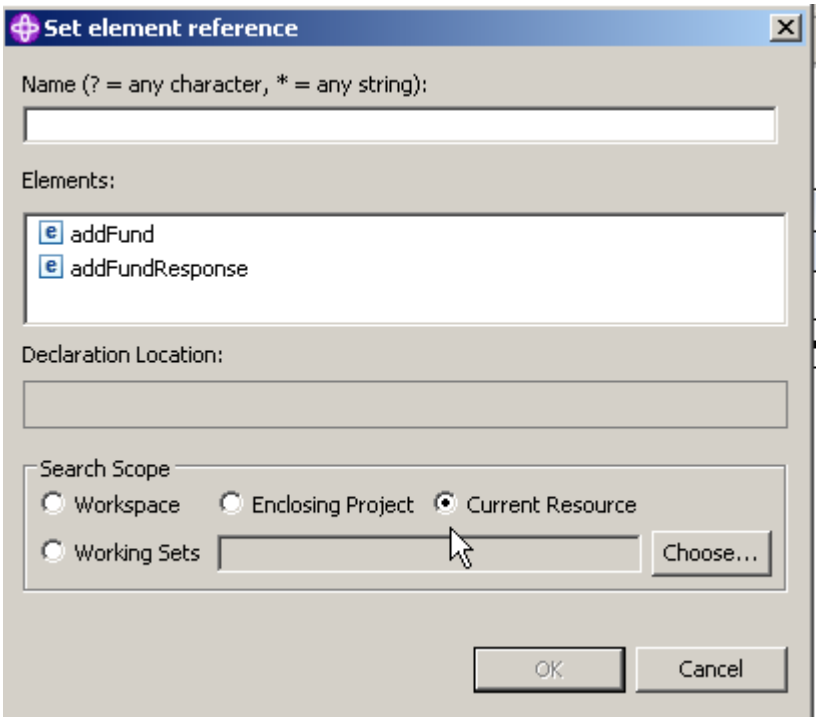
38.Click on the NewPart element next to the input element to make it editable



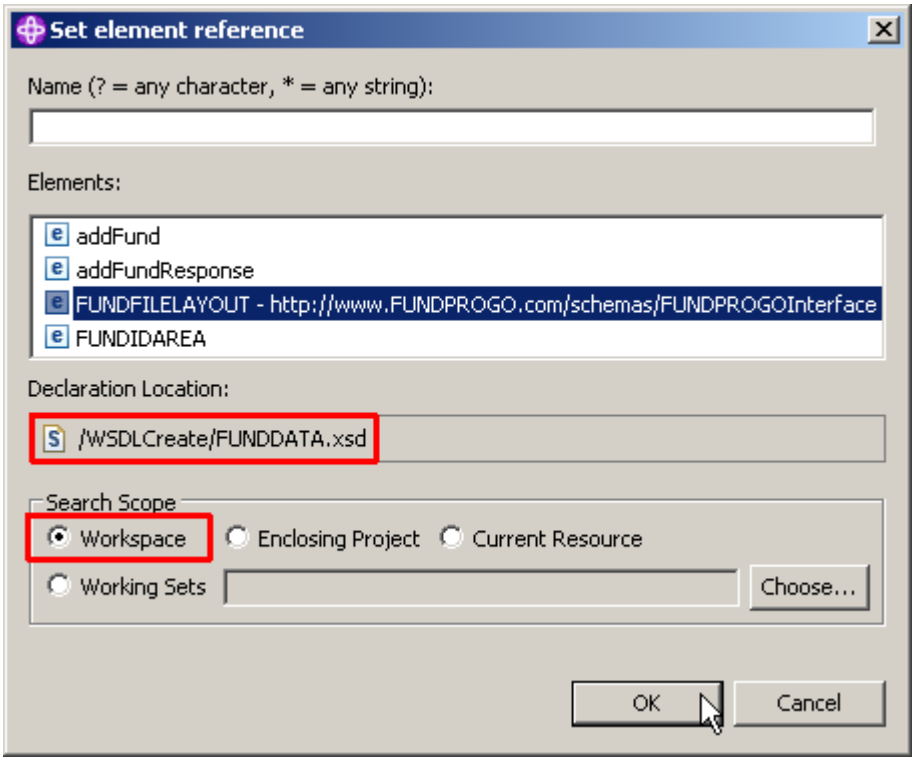
and rename it to FundData.



Now right-click on the FundData element and from the context menu, select Set Element → Existing Element...



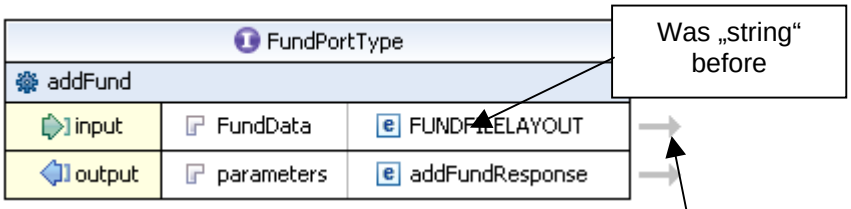
39.On the Set element reference panel, first of all, change the Search Scope from Current Resource to Workspace.



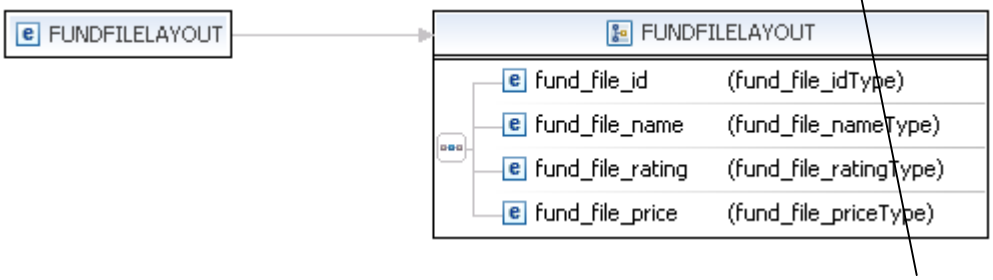
Then you will find two new elements, FUNDFILELAYOUT and FUNDIDAREA.

Choose FUNDFILELAYOUT (the text will expand) and ensure, that the Declaration Location is the /WSDLCreate/FUNDDATA.xsd as shown above.

Finally, click the OK button.

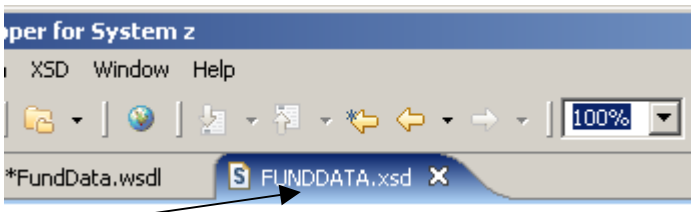


40.Your FundPortType should look like this:

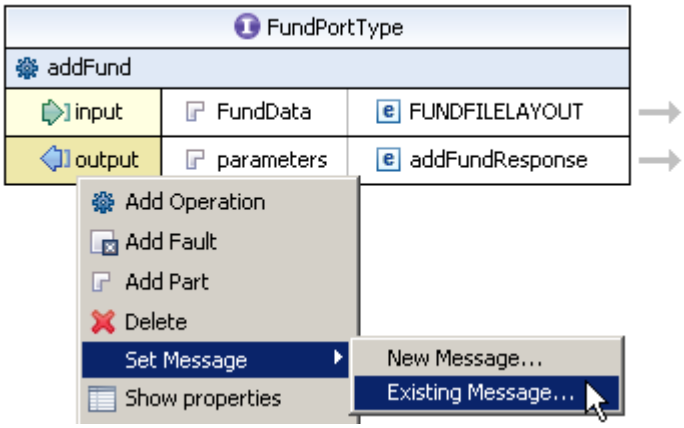


41.Let's have a look at the FUNDFILELAYOUT element. If you click on the arrow next to it, the FUNDDATA.xsd will open.

As you can see below, FUNDFILELAYOUT is a ComplexType consisting of four other elements.

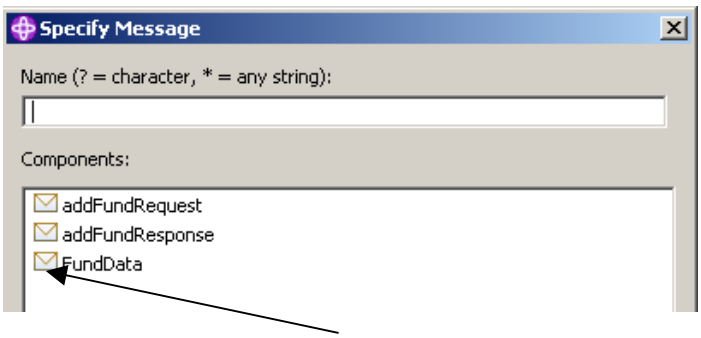


When you're finished, close the FUNDDATA.xsd.

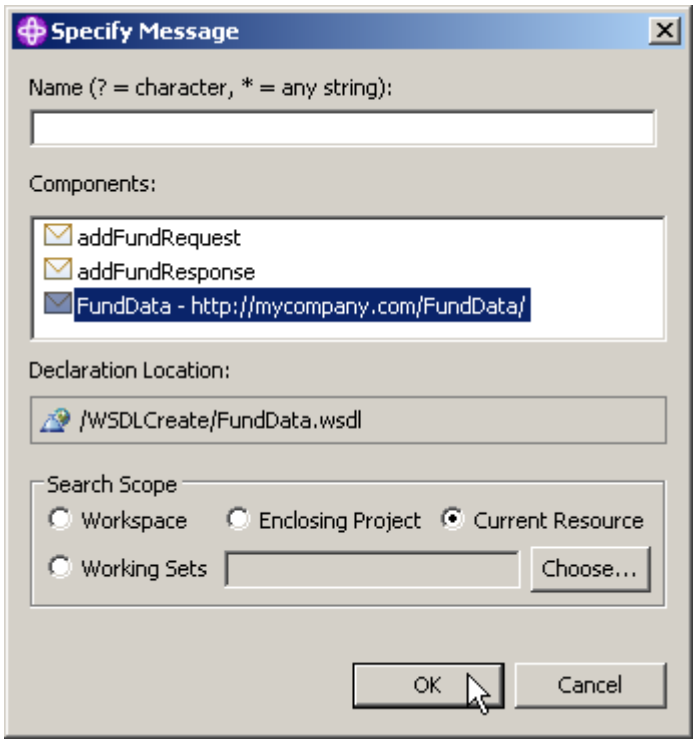


42.Now we have to configure the layout of the outgoing messages. As we use the same layout for incoming and outgoing messages, this is an easy task because we already configured everything we need.

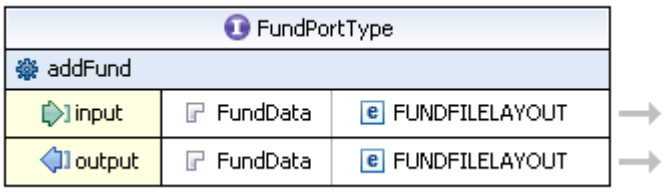
Right-click the output element and select Set Message → Existing Message...



43.In the Specify Message dialog, select the earlier created FundData Message and ….



and click OK.

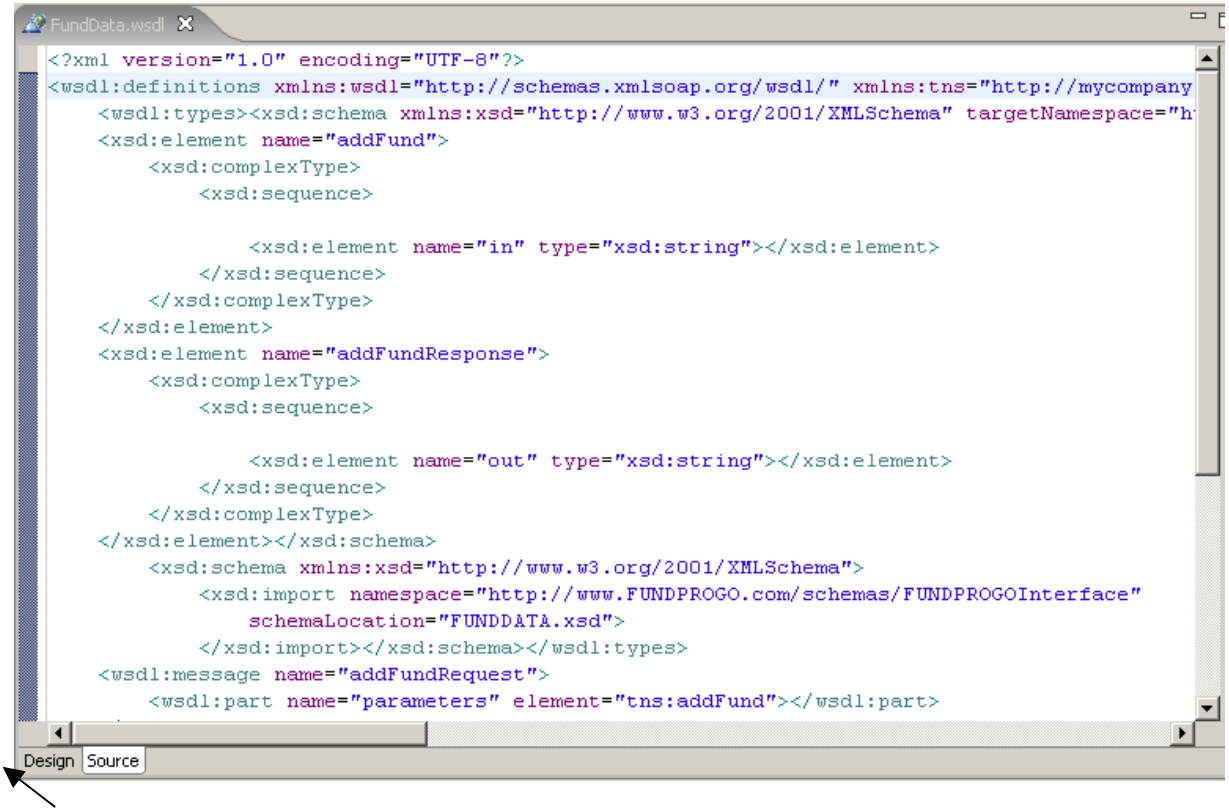


44.Your FundPortType should now look like that:

Note: You could have set the layout for the output message as well with the help of the Properties and Outline view. At that point, you will note, that you can specify a name for both, the input and the output message. If you feel it is important to you, you can do it now. You won't see a change in the Design view though (Only in the Source view!).

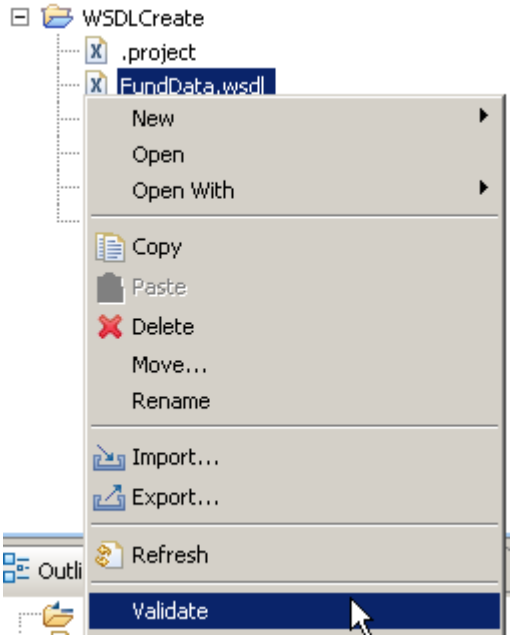
45.We finished configuring our message layout. Both, input and output messages use the same layout.

Press CTRL + S to save your work in the FundData.wsdl editor (the * next to FundData.wsdl should disappear).



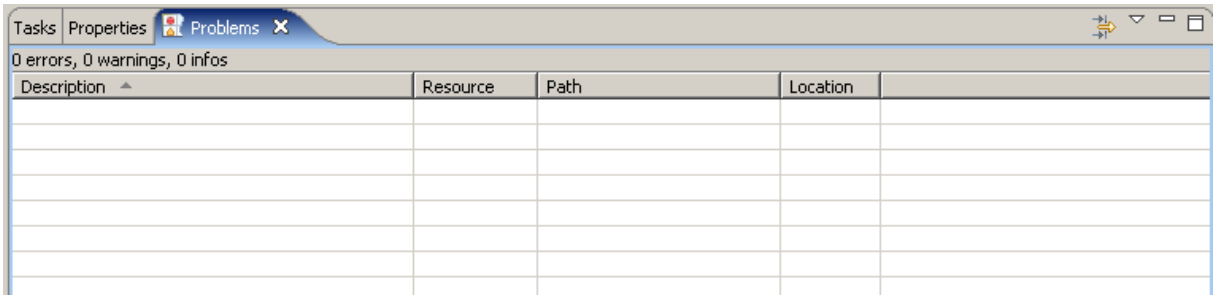
Click on Source to have a view of the generated XML code. Remember, a WSDL file consists of XML code.

If you would have to code this manually, this would be a very error prone process. One of XML's strong characteristics is, that using modern tools, you just about never have to handcode it.



46.As a final step, you should validate your WSDL file.
From the Navigator view, right-click FundData.wsdl and select Validate from the context menu.

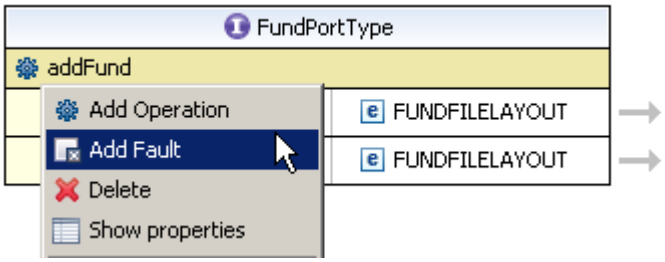
If you have a valid WSDL file, you won't get a popup or similar. In addition, there should be no errors in the Problems view. If the Problems view is not enabled, click Window → Show View → Problems on the menu bar to enable it.



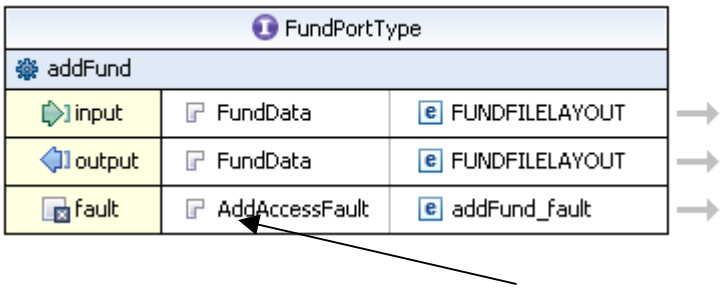
Note: You MUST have an aktive connection to the internet to validate the file because a WS-I Test Assertion Document (TAD) document is mandatory! Otherwise you will get an error like: „The WS-I Test Assertion Document (TAD)document was either not found or could not be processed“.
So check if you need to configure the use of a proxy,
for instance In Window → Preferences → Internet → Proxy Settings.

2.8 Add a fault element to the addFund Operation

If a fund cannot be added, the operation will throw a fault. In this section we will add a fault to the addFund operation.



47.Right klick on Design again to leave the Source representation. In the opened FundData.wsdl file, right-click the addFund operation, and select Add Fault.



48.You may change the name ot the fault part element from parameters to AddAccessFault.

The fault message was added with the part AddAccessFault and the element addFund_fault. The part is of type string. This a typical return type for most faults.

49.Press CTRL + S to save your WSDL file.

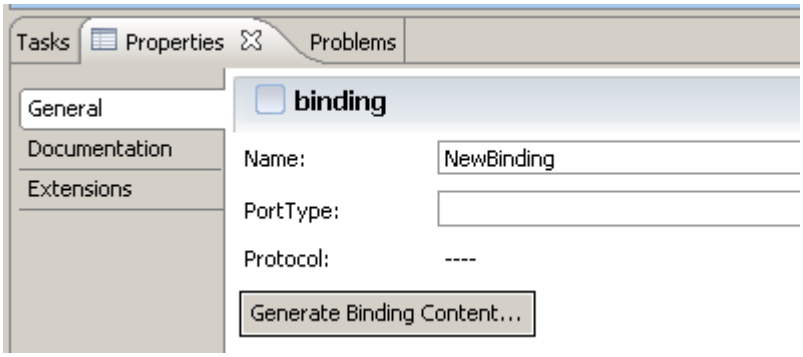
50.You should validate you edited WSDL-file again like you did above (see sectio 46).
If you get no errors, proceed to the next chapter.

2.9 Add Binding Information to the WSDL File

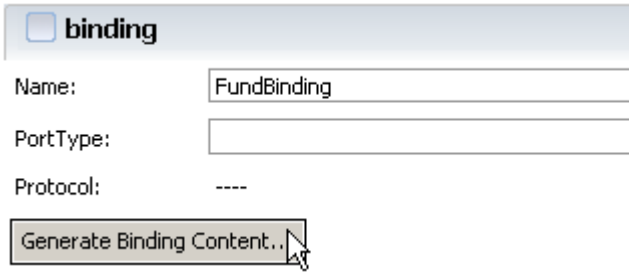
We have added a Port Type, an operation, and messages. These are generic and could apply to any service (i.e. these are transport independent). In this section, we will tell the WSDL editor that we are describing a Web Service. A Web Service is SOAP over HTTP. This means that we will be adding a SOAP Binding. In a later Part (Part 9) of this Tutorial we will add Service information that will indicate where the Web Service is located.

51.Right-click on an empty spot in your opened WSDL-file and select Add Binding from the context menu.

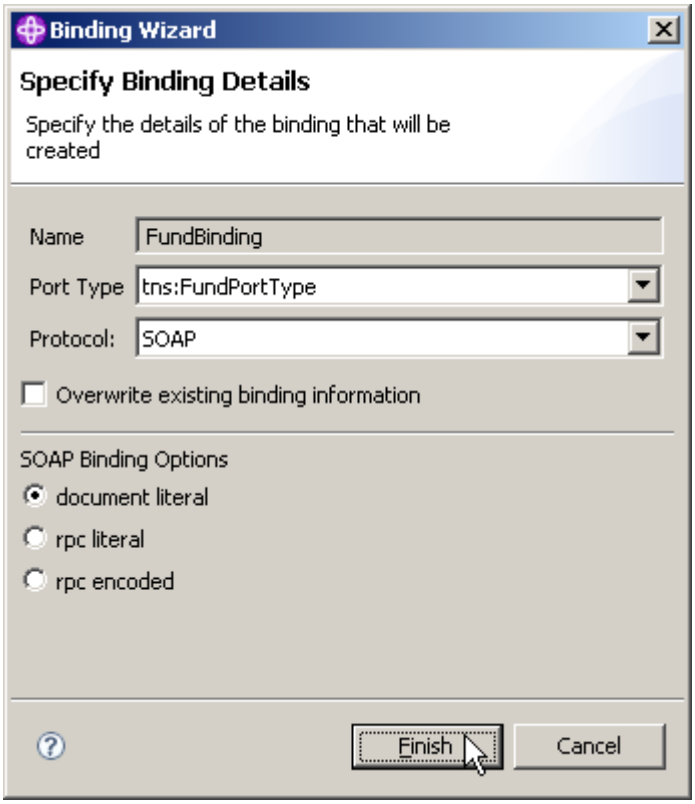
52.You will now see a Binding Element like that: 



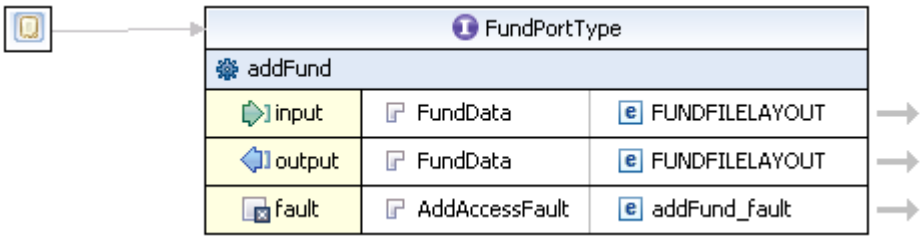
53.Click the element. In the Properties view, change the name from NewBinding



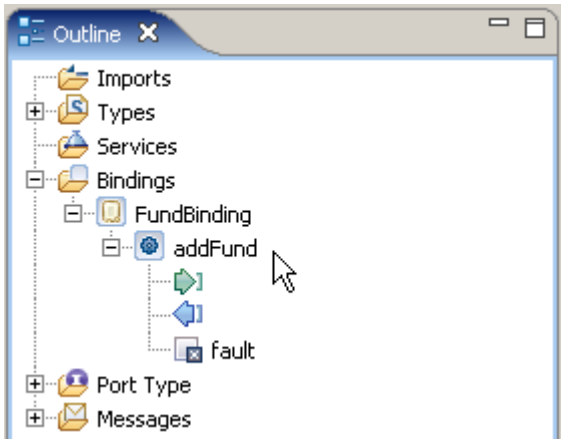
to FundBinding. Then, click the Generate Binding Content.button.



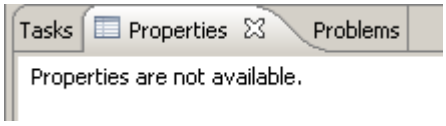
54.In the Specify Binding Details dialog, change the Port Type to tns:FundPortType and the Protocol to SOAP. Leave the defaults for the SOAP Binding Options.
Click Finish.



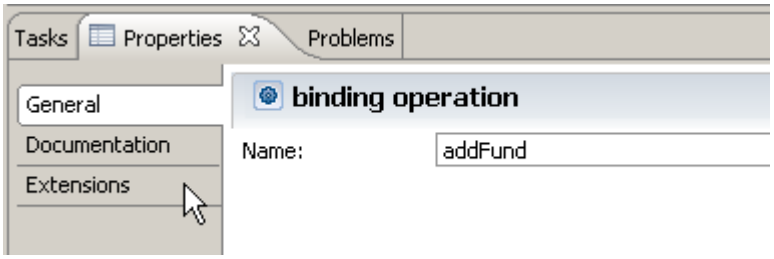
55.Your FundData.wsdl should now look similar as shown above.



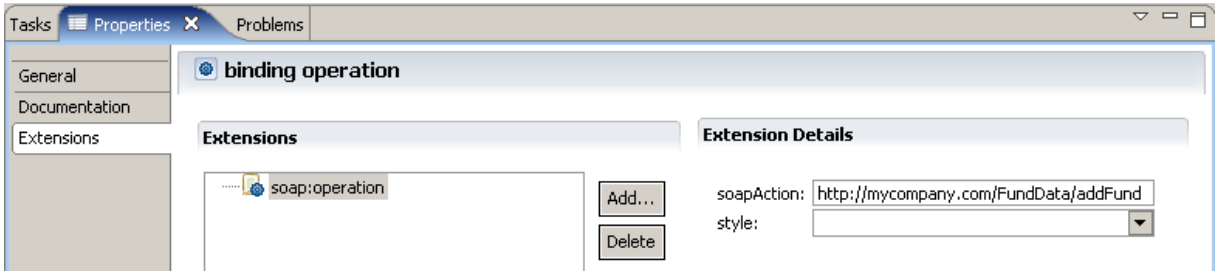
56.Fully expand the Bindings tree structure in the Outline view. You will note, that your newly created FundBinding has a binding operation named addFund. This indicates, Port Type and Binding are connected now.



At this point, the Properties View is empty



In the Outline view, click on addFund and have a look at the Properties view. Switch to Extension in the menu on the left and …



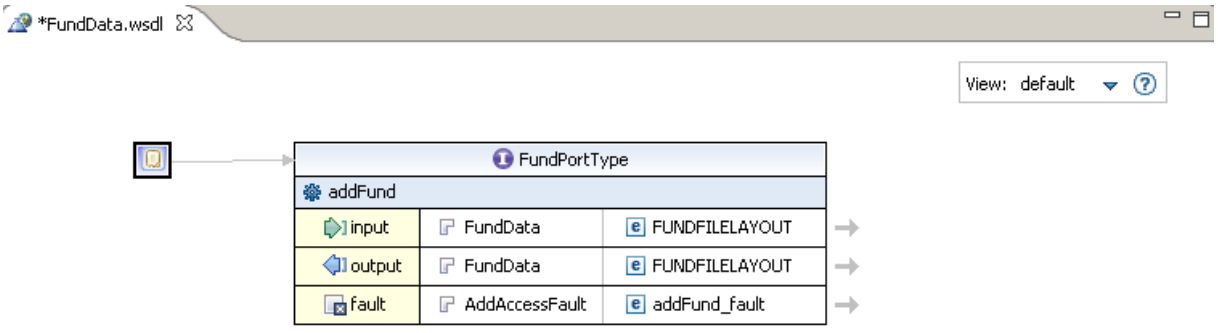
ensure that under Extension Details, soapAction is http://mycompany.com/FundData/addFund and

style is document.

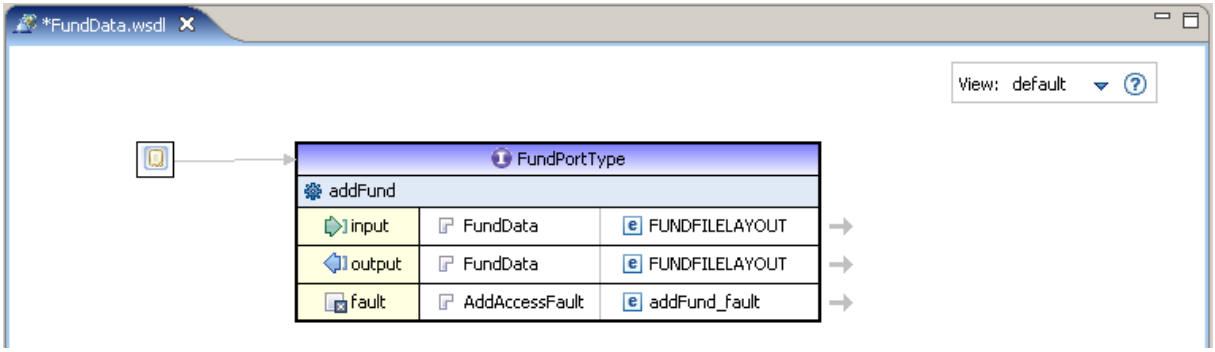
57.In the FundData.wsdl and the Properties View press CTRL + S to save your changes. Now validate your file again (section 46). If there are no errors, proceed to the next chapter.

2.10 Add Service Information

We have specified what business operation is available and the information that is to be sent and received by the operation. We specified that the information would be sent using SOAP, and now we need to specify where the Web Service resides.



58In your opened FundData.wsdl-file,

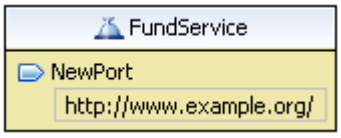


highlight FundPortType, and

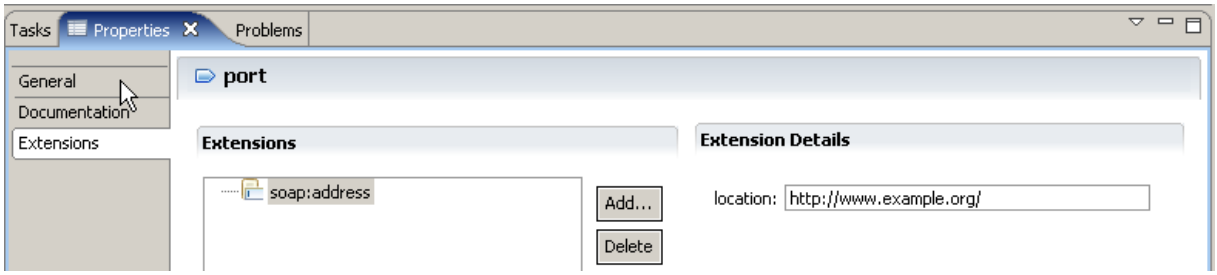
and select Add Service from the context menu.

59.A new Service Element will appear.

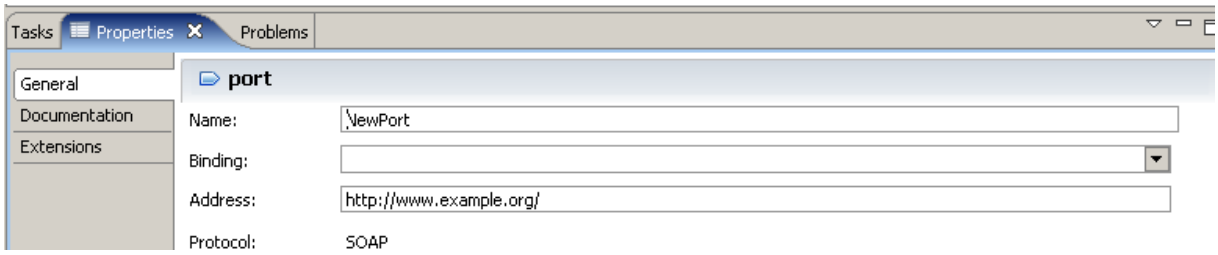
Click on the name of the new Element and change it from NewService to FundService.



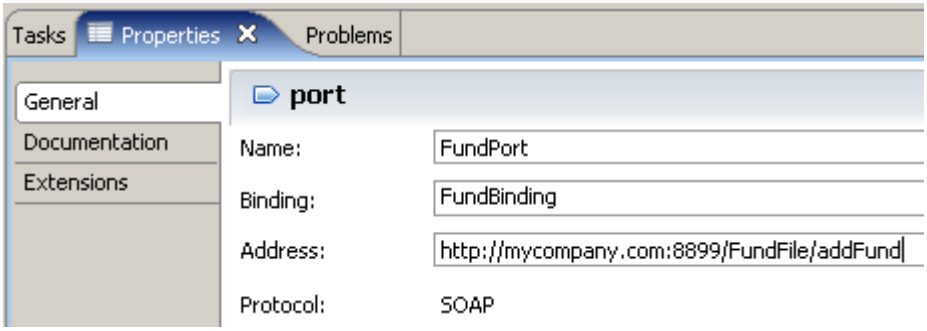
60.Click on the NewPort element. The colour changes.



In the Properties view click on General

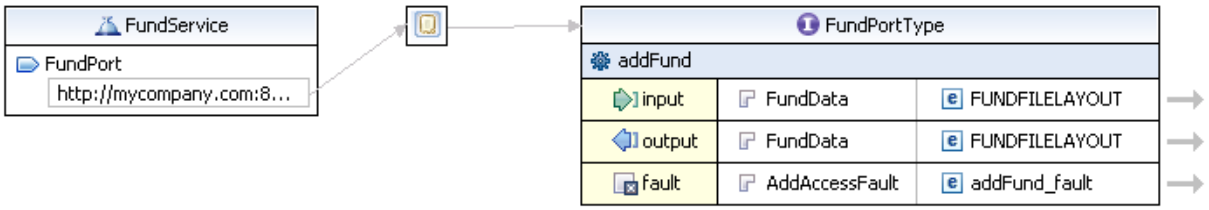


Now



specify

- **as Name: FundPort**
- **as Binding: FundBinding**
- **as Address: http://mycompany.com:8899/FundFile/addFund**



Your FundData.wsdl should now look like this:

Note: The Address is also known as the Endpoint of a Web Service.

61.Press CTRL + S to save the file and validate it a last time (see sectio 46).

62.If there are no errors, close the WSDL-file.

Now you have a fully functional WSDL document that describes a business operation (addFund), input and output messages and their layout (both point to the FUNDDATA.xsd), a fault (the service will return a fault if something goes wrong), a binding (SOAP over HTTP), and a service (which specifies the location of the endpoint).

It is a valid WSDL document, and it adheres to the WS-I basic profile 1.1. WS-I is an open industry organization chartered to provide Web Services interoperability across platforms, operating systems, and programming languages.

Congratulation, you successfully completed the Web Services Tutorial01 !

2.11 Create more WSDL Files (Optional)

This is an optional part that will give you more practice with WSDL and the WDz WSDL editor. The backend program that will be invoked supports Fund Create, Read, Update, and Delete processing. We have just created some WSDL that defines an addFund operation that will invoke Fund Create on the backend program. In this optional step, you can create a WSDL file for each of the getFund, deleteFund, and updateFund operations.

You probably also noticed that in Part 3 we made two XSDs: one that described the fund file layout (FUNDDATA.xsd), and one that described the fund id (FUNDID.xsd). For the addFund operation, both the input and the output of the operation was FUNDDATA.xsd. This means that so far, you haven't used the FUNDID.xsd. If you do this part of the tutorial, you will use the FUNDID.xsd for the input message for the getFund operation, and for the deleteFund operation.

The directions in this part of the lab are not as detailed as the directions in the early part of this lab so you may need to look back at a few earlier steps if the directions are not explicit enough.

Add WSDL files (one each) for the following operations. These operations use the FUNDID.xsd and FUNDDATA.xsd files as indicated in the table below.

Operation	input	output	fault
getFund	FundId	FundData	GetAccessFault
deleteFund	FundId	FundData	DeleteAccessFault
updateFund	FundData	FundData	UpdateAccessFault

Be sure to save and validate your WSDL files.

Resources

Webtut01.zip contains resources as a zip Archiv. Download at

www.cedix.de/Vorles/Band3/Resources/webtut01.zip