Objectives

- At the end of this chapter you will be able to Understand following concepts in web services
  - Message Part
  - Operation
  - Port Type
  - Binding
  - Port
  - Target namespace
- The difference between RPC style and document style web services
- Introduction to WSDL, a web service interface

Cross platform operations across the Internet:- Requirement

- **Requirement:-**
  - We'd like to provide a service to the public or to some business partners:
    - They can send us two strings and we will concatenate them and return the string.
    - Of course, in the real world we provide a more useful service.

Cross platform operations across the Internet :- Requirement

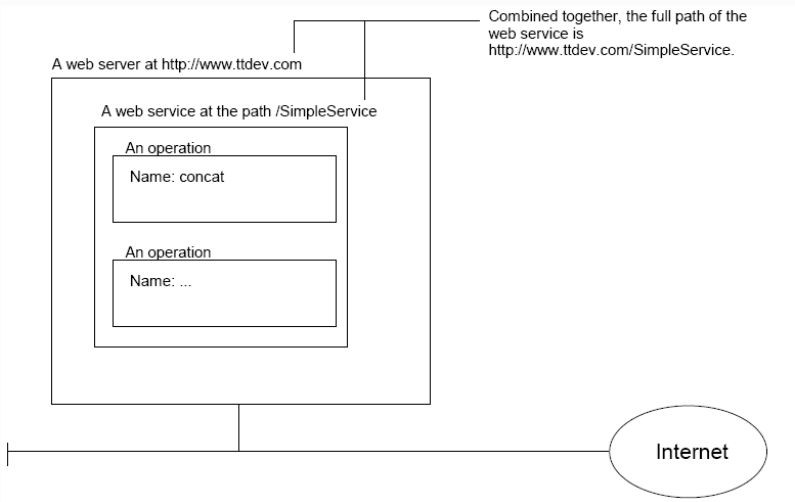
- **Requirement:-**
  - There are several major requirements:
    - The users may be using different languages (Java, C# and etc.) and using different platforms (Windows, Linux and etc.).
    - Our service must be accessible by different languages and platforms.
    - Users will call our service across the Internet and there may be firewalls in between.
    - Our service must be able to go through firewalls.

Cross platform operations across the Internet :- Probable analysis

- **After analyzing the requirement the probable solution might be**
  - A "web service".
  - For example, we may make a web service accessible on the *host www.ttdev.com* and accessible as */SimpleService*, so the full URL is *http://www.ttdev.com/SimpleService*. This is called the "**endpoint**" of the web service
  - Our web service may support one or more operations. One ***operation*** may be named "*concat*"

seed
beyond the obvious

Cross platform operations across the Internet :- Probable analysis



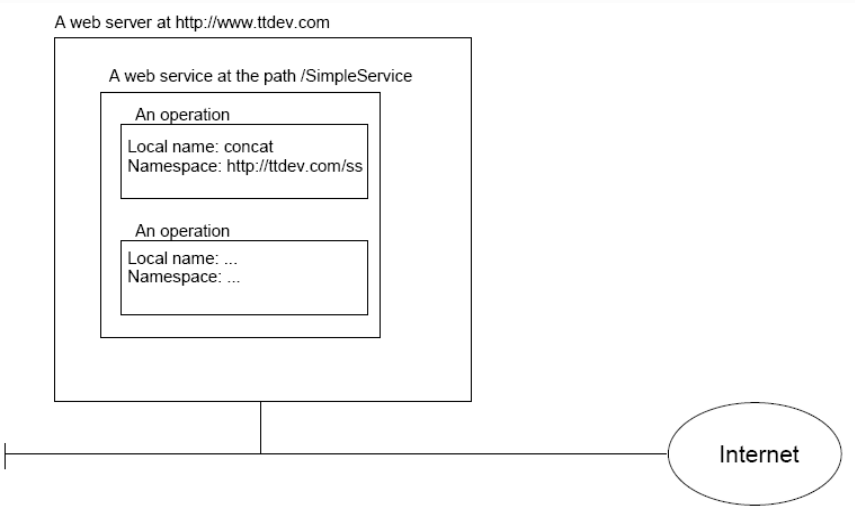
seed
beyond the obvious

Cross platform operations across the Internet :- Probable analysis

- We'd like to provide a *globally unique* name to each operation so that we can have our "concat" operation while another person may have his/her "concat" operation.
- So, in addition to the name, we may declare that the "concat" name above is in the "***namespace***" of http://ttdev.com/ss

seed
beyond the obvious

Cross platform operations across the Internet :- Probable analysis



seed
beyond the obvious

Cross platform operations across the Internet :- Probable analysis

- A namespace is just like a Java package, but it is not in a dot format like *com.ttdev.foo;* it is in the format of a URL. So, the full name of the operation will be *"concat" in namespace http://ttdev.com/ss*.
- The name "concat‟ is called the "*local name*". The full name is called a "*QName (qualified name)*"

seed
beyond the obvious

Cross platform operations across the Internet :- Probable analysis

- We may wonder what this http://ttdev.com/ss namespace means??
- The answer is that it has no particular meaning. Even though it is a URL, it does NOT mean that we can use a browser to access this URL to get a web page (if we do, we may get a file not found error).
- The only important thing is that it must be globally unique.

seed
beyond the obvious

Cross platform operations across the Internet :- Probable analysis

- Note that the namespace is a completely different concept from the endpoint.
- The endpoint really is the location, while the namespace is just a unique id.
- We could easily move the web service to another web server and thus it will have a different endpoint, but the namespaces of its operations will remain unchanged.

seed
beyond the obvious

Cross platform operations across the Internet :- RPC style web service

- Our concat operation may take two parameters. One is named "s1" and is a string. The other is named "s2" and is also a string. The return value is also a string

An operation

```
Local name: concat
Namespace: http://ttdev.com/ss
Parameters:
  s1: string
  s2: string
Return:
  string
```


seed
beyond the obvious

Cross platform operations across the Internet :- RPC style web service

An operation

```
Local name: concat
Namespace: http://ttdev.com/ss
Parameters:
  s1: string
  s2: string
Return:
  string
```

- what does the above "*string*" type mean?? Is it the Java string type??
- No, we can't say that because it must be language neutral.

seed
beyond the obvious

Cross platform operations across the Internet :- RPC style web service

An operation

```
Local name: concat
Namespace: http://ttdev.com/ss
Parameters:
  s1: string
  s2: string
Return:
  string
```

- Fortunately, the XML schema specification defines some basic data types including a *string* type.
- Each of these data types has a QName as its id. For example…

seed
beyond the obvious

Cross platform operations across the Internet :- RPC style web service

An operation

```
Local name: concat
Namespace: http://ttdev.com/ss
Parameters:
  s1: string
  s2: string
Return:
  string
```

Data type	*Local name*	*namespace*
string	string	http://www.w3.org/2001/XMLSchema
integer	int	http://www.w3.org/2001/XMLSchema
...	...	...

seed
beyond the obvious

Cross platform operations across the Internet :- RPC style web service

- So, the interface of our operation should be written as:

An operation

```
Local name: concat
Namespace: http://ttdev.com/ss
Parameters:
  s1: string in http://www.w3.org/2001/XMLSchema
  s2: string in http://www.w3.org/2001/XMLSchema
Return:
  string in http://www.w3.org/2001/XMLSchema
```


seed
beyond the obvious

Cross platform operations across the Internet :- RPC style web service

- Actually, in web services, a method call is called an "*input message*" and a parameter is called a "*part*". The return value is called an "*output message*" and may contain multiple parts. So, it is more correct to say:

An operation

```
Local name: concat
Namespace: http://ttdev.com/ss
Input message:
  Part 1:
    Name: s1
    Type: string in http://www.w3.org/2001/XMLSchema
  Part 2:
    Name: s2
    Type: string in http://www.w3.org/2001/XMLSchema
Output message:
  Part 1:
    Name: return
    Type: string in http://www.w3.org/2001/XMLSchema
```

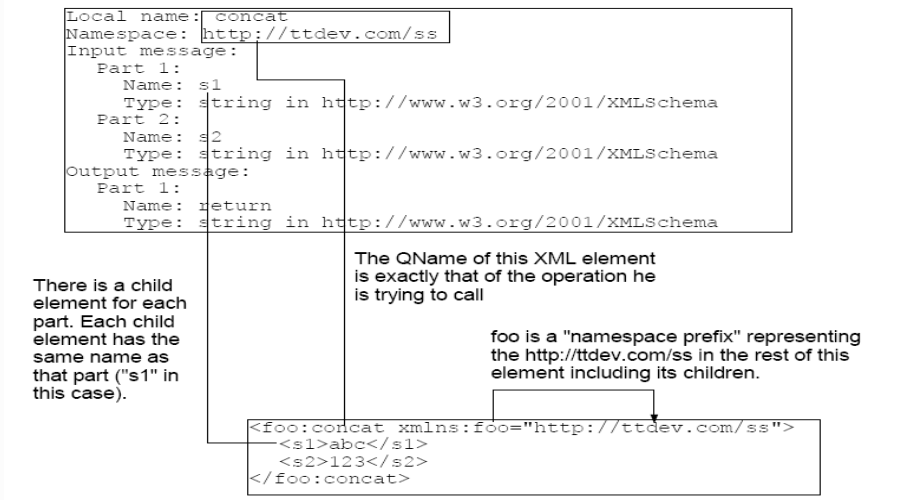

seed
beyond the obvious

Cross platform operations across the Internet :- RPC style web service

- When someone calls this operation, he/she can send us an XML element as the input message like…..

seed
beyond the obvious

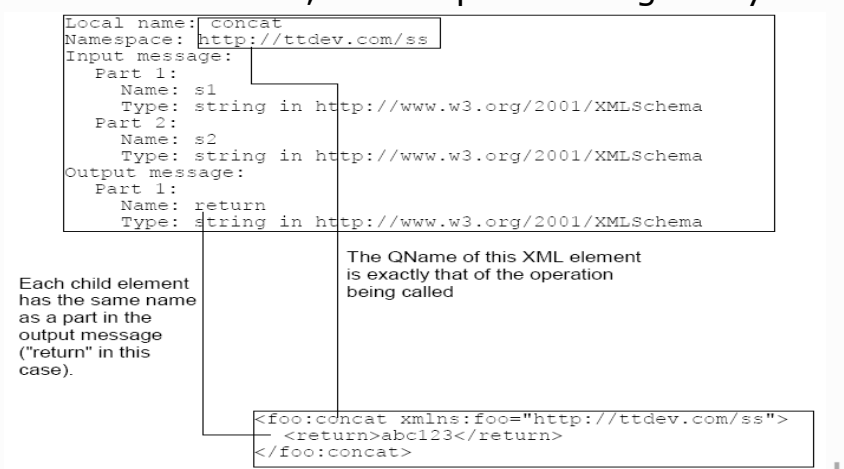
Cross platform operations across the Internet :- RPC style web service



seed
beyond the obvious

Cross platform operations across the Internet :- RPC style web service

- When we return, the output message may be



seed
beyond the obvious

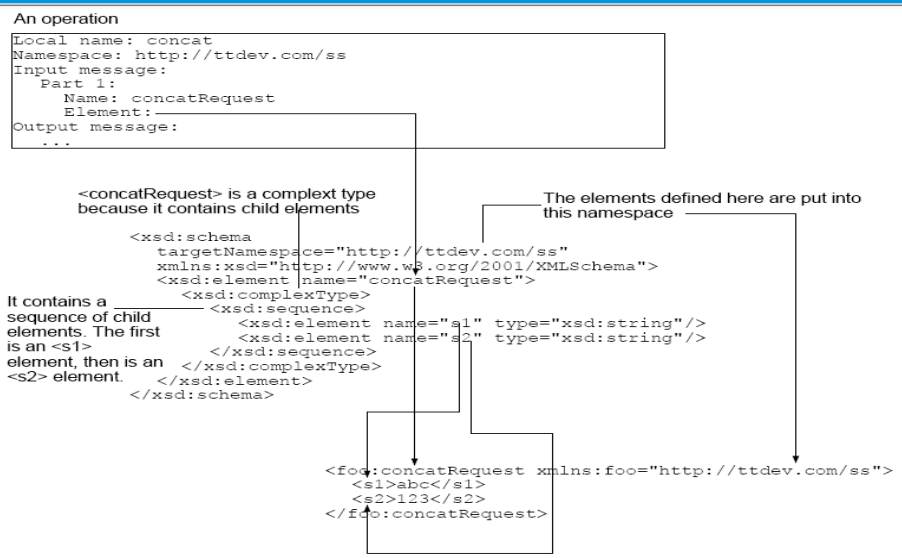
Cross platform operations across the Internet :- RPC style web service

- This kind of web service is called "***RPC style***" web service (RPC stands for "Remote Procedure Call").
- That is, the operation QName and the names of the parts are used to create the input and output messages.

Cross platform operations across the Internet :- Document style web service

- The *RPC-style* is not the only way we design the interface of our web service.
- For example, we may say that its input message only contains a single part which is an element defined in a schema.
- In that schema, it is defined as an element named "*concatRequest*" that contains two child elements <s1> and <s2>:

Cross platform operations across the Internet :- Document style web service



seed
beyond the obvious

Cross platform operations across the Internet :- Document style web service

- Note that the schema is included in the interface of our web service:

seed
beyond the obvious

Cross platform operations across the Internet :- Document style web service

A web service

A schema

```
<xsd:schema
   targetNamespace="http://ttdev.com/ss"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <xsd:element name="concatRequest">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="s1" type="xsd:string"/>
            <xsd:element name="s2" type="xsd:string"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

An operation

```
Local name: concat
Namespace: http://ttdev.com/ss
Input message:
   Part 1:
      Name: concatRequest
      Element: concatRequest in http://ttdev.com/ss
Output message:
   ...
```

seed
beyond the obvious

25

Cross platform operations across the Internet :- Document style web service

- As we can see, a part may be declared as a particular element (<concatRequest> defined in our schema) or as any element having a particular type (string defined in XML schema specification). In either case it is identified using a QName

seed
beyond the obvious

26

Cross platform operations across the Internet :- Document style web service

- When someone calls this operation, he/she will send us a <concatRequest> element as the input message like:

```
<foo:concatRequest xmlns:foo="http://ttdev.com/ss">
  <s1>abc</s1>
  <s2>123</s2>
</foo:concatRequest>
```


seed
beyond the obvious

Cross platform operations across the Internet :- Document style web service

- Similarly, for the output message, we may specify that it contains only one part and that part is a <concatResponse> element like…...

seed
beyond the obvious

Cross platform operations across the Internet :- Document style web service

Cross platform operations across the Internet :- Document style web service

- This kind of web service is called "***document style***" web service. That is, the input message will contain a single part only which is well defined in a schema. The same is true of the output message.
- If we go back to check the input message for the RPC style service, it should be revised as:

Cross platform operations across the Internet :- RPC style web service

```
<foo:concat>
  xmlns:foo="http://ttdev.com/ss"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-Instance">
  <s1 xsi:type="xsd:string">abc</s1>
  <s2 xsi:type="xsd:string">123</s2>
</foo:concat>
```

This attribute is used to explicitly state the XML data type of the body of an element ("abc" here). This is useful when the element (<s1>) itself is not defined in a schema. This "type" attribute is defined in the http://www.w3.org/2001/XMLSchema-Instance namespace, so you need to introduce a prefix for it:

Now, let's compare the input messages of the RPC style web service and the document style web service:

seed
beyond the obvious

Cross platform operations across the Internet :- Document style web service

RPC style	`<foo:concat>`
	`  xmlns:foo="http://ttdev.com/ss"`
	`  xmlns:xsd="http://www.w3.org/2001/XMLSchema"`
	`  xmlns:xsi="http://www.w3.org/2001/XMLSchema-Instance">`
	`  <s1 xsi:type="xsd:string">abc</s1>`
	`  <s2 xsi:type="xsd:string">123</s2>`
	`</foo:concat>`

Document style	`<foo:concatRequest xmlns:foo="http://ttdev.com/ss">`
	`  <s1>abc</s1>`
	`  <s2>123</s2>`
	`</foo:concatRequest>`

seed
beyond the obvious

Cross platform operations across the Internet :- Document style web service

- Not much difference, right? The significant difference is that
- The former can't be validated with a schema while the latter can.
- Therefore, document style web service is becoming the dominant style.
- According to an organization called "*WS-I (web services interoperability organization)*", we should use document style web services only.

seed
beyond the obvious

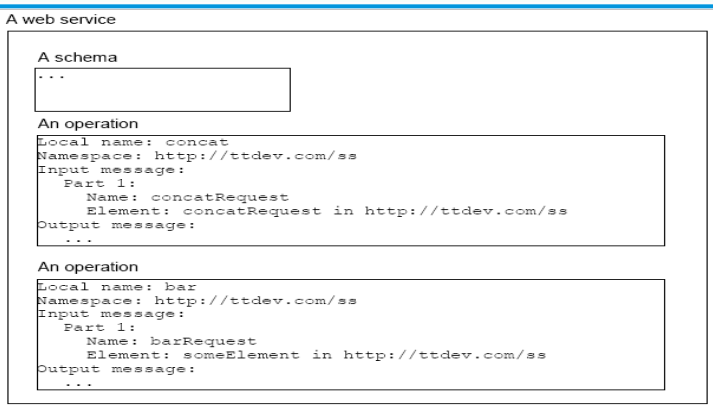
Determining the operation for a document style web service

```
<foo:concatRequest xmlns:foo="http://ttdev.com/ss">
   <s1>abc</s1>
   <s2>123</s2>
</foo:concatRequest>
```

- To call an operation in a document style web service, one will send the single part of the input message only.
- Note that it does NOT send the operation name in any way.
- Then if there are more than one operations in the web service, how can it determine which one is being called??

seed
beyond the obvious

Determining the operation for a document style web service

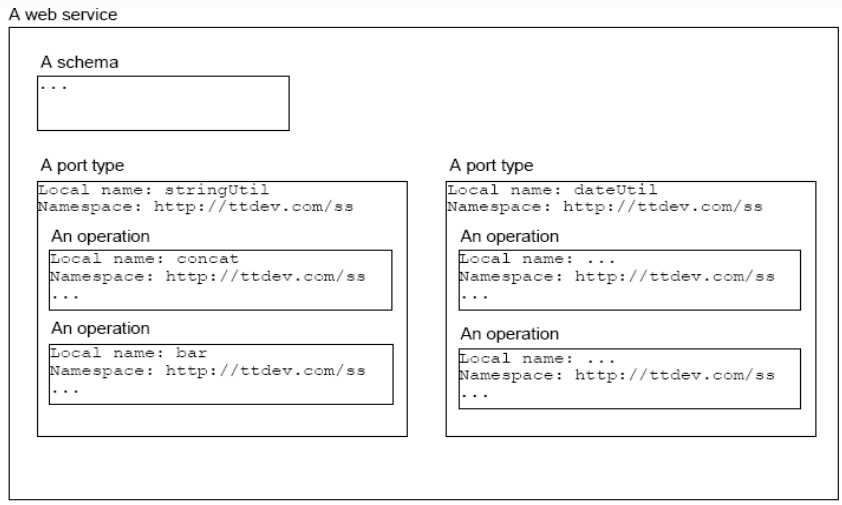


- In that case it will see if the input message is a <concatRequest> or a <someElement> to determine.
- What if both take a <someElement>?? Then it is an error and it won't work

Port Type

- Actually, a web service doesn't directly contain a list of operations. Instead, operations are grouped into one or more "***port types***".
- A port type is like a Java class and each operation in it is like a static method.
- For example, in the web service we are discussing, we could have a port type named "*stringUtil*" containing operations for strings, while having another port type named "*dateUtil*" containing operations for dates. The name of a port type must also be a QName

Port Type



seed
beyond the obvious

Binding

- Actually, a port type may allow us to access it using different message formats.
- The message format that we have seen is called the "***Simple Object Access Protocol (SOAP)***" format.
- It is possible that, say, the *stringUtil* port type may also support a plain text format:

```
concat(s1='abc', s2='123')
```

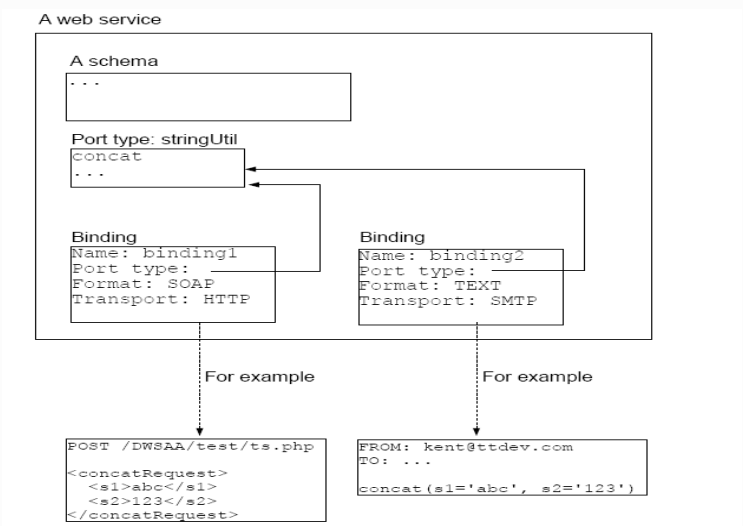

seed
beyond the obvious

Binding

- In addition to the message format, a port type may allow the message to be carried (***transported***) in an HTTP POST request or in an email.
- Each supported combination is called a "binding":

seed
beyond the obvious

Binding



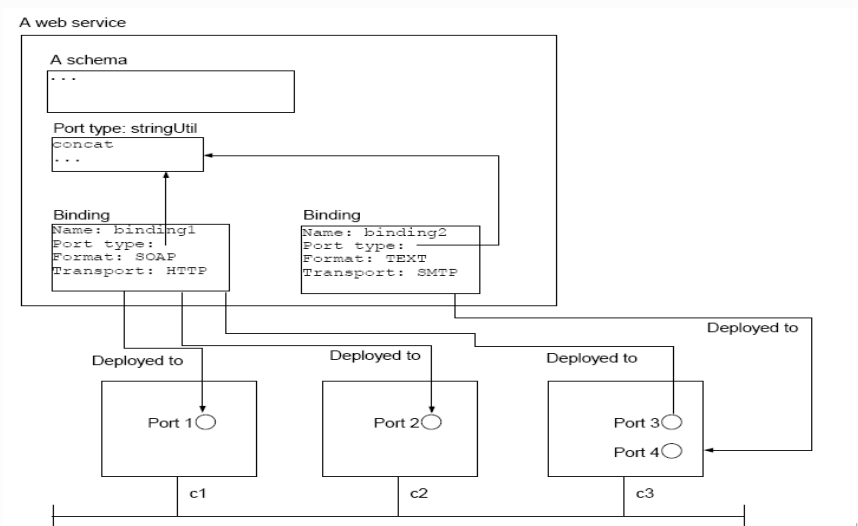
seed
beyond the obvious

Port

- Suppose that there are just too many people using our web service, we decide to make it available on more than one computers
- For example, we may deploy the above binding 1 on computers c1, c2 and c3 and deploy binding 2 on c3.
- In that case it is said that we have four ***ports*** Three ports are using binding 1 and one using binding 2:

seed
beyond the obvious

Port



seed
beyond the obvious

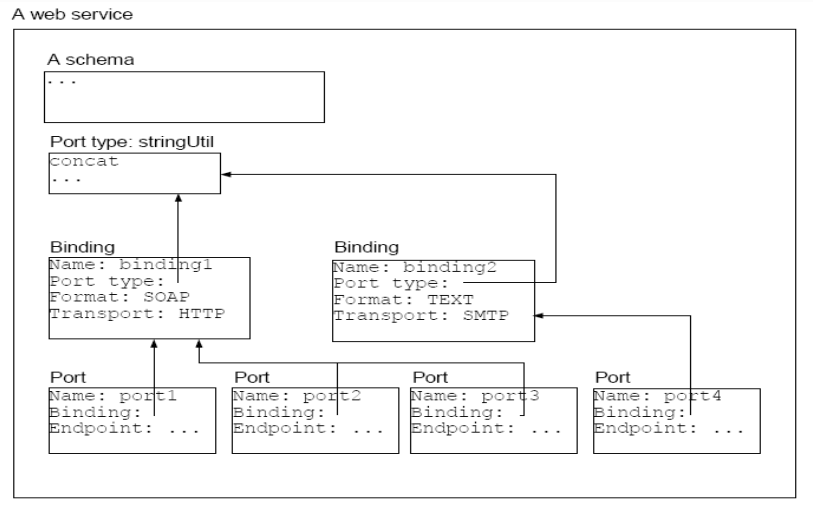
Port

- Note that it does NOT mean that the requests received by these three computers will be forwarded to a computer hiding behind for processing.
- Instead, it means that there is some software implementing the port type installed on these three computers.

Port

- There is no need that the same piece of software is installed onto the different computers.
- For example, on c1, port 1 may be written in Java, while on c2, port 2 may be written in C#.
- The important point is that they both support the operations specified in port type stringUtil and the message format and transport specified in the binding 1.
- Port 4 must also implement the same operations too (same port type) but the message format and transport are different.
- To tell others about this arrangement, you include these ports in the interface of the web service:

Port



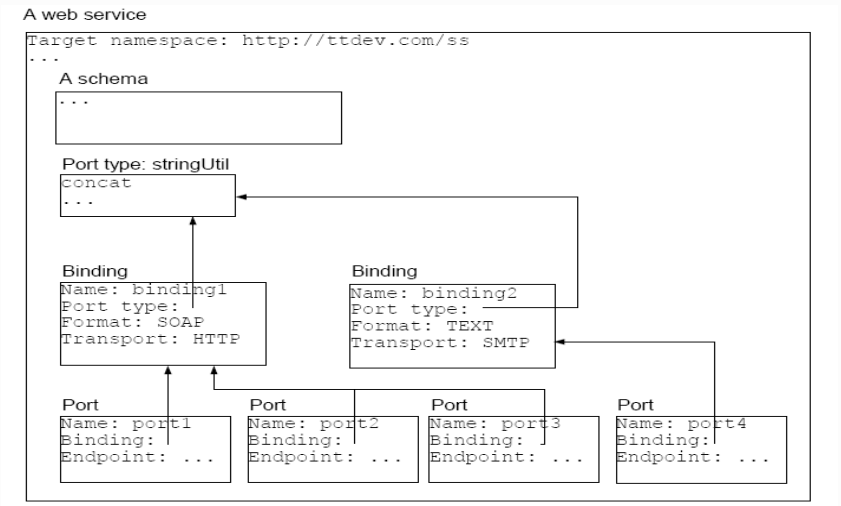
seed
beyond the obvious

Target namespace

- We have been using the same namespace for the operation names, port type names and etc. in this web service. Do they have to be in the same namespace??
- By default, this is the case: There is a single namespace for a web service to put the names into. This is called the "*target namespace*" for the web service:

seed
beyond the obvious

Target namespace



seed
beyond the obvious

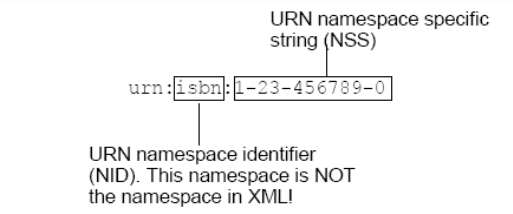
Target namespace

- Basically a namespace is good as long as it is globally unique
- A namespace must be a URI. URI stands for Uniform Resource Identifier.
- There are two kinds of URI.
  - One is *URL* such as http://www.foo.com/bar.
  - The other is *URN*. A URN takes the format of urn:<some-object-type>:<someobject-id>

seed
beyond the obvious

Target namespace

- An XML namespace must be a URI. We can use a URL or a URN.
- Functionally there is no difference at all. For example, we may use say urn:ttdev.com:ss as the target namespace for our web service instead of http://ttdev.com/ss without changing any functionality.



49

seed
beyond the obvious

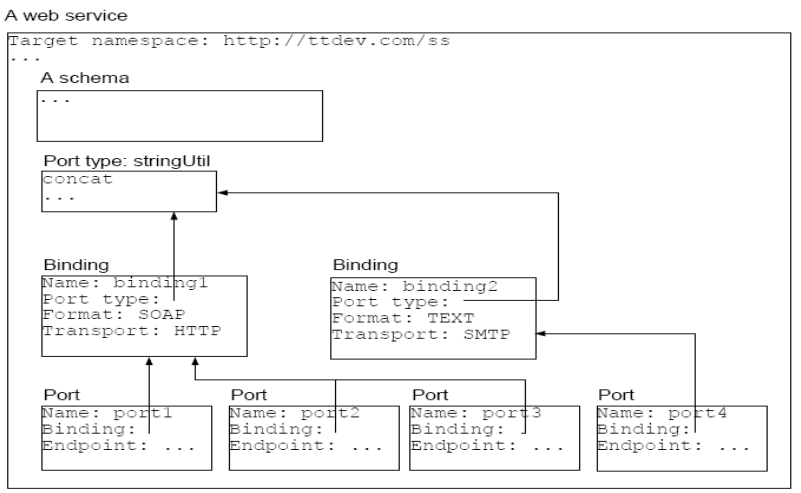
WSDL

- By now we have finished designing the interface for our web service:
- It fully describes our web service. This description language (terms and concepts) is called "WSDL (Web Services Description Language)".

50

seed
beyond the obvious

WSDL



seed
beyond the obvious

Quick Recap . . .

- A web service is platform neutral, language neutral and can be accessed across the Internet.
- A web service has one or more ports. Each port is a binding deployed at a certain network address (endpoint).
- A binding is a port type using a particular message format and a particular transport protocol. A port type contains one or more operations.
- An operation has an input message and an output message. Each message has one or more parts.
- Each part is either a certain element defined in the schema of the web service, or any element belonging to a certain element type in that schema.
- All above information is fully described in WSDL.

seed
beyond the obvious

Quick Recap . . .

- To call a RPC style web service, one will create an XML element with the name of the operation and a child element for each of its input message part.
- To call a document style web service, one will just send the one and only part of its input message.
- Because the XML element used to call a RPC style web service is not defined in any schema, for better interoperability, one should create document style web services.