Web Services

SET09103 Advanced Web Technologies

School of Computing
Napier University, Edinburgh, UK
Module Leader: Uta Priss

Outline

Introduction

PHP example

Web services

- Accessing remote functions
- RPC: remote procedure calls
- Web APIs
- Server/client can use different programming languages
- industry- and vendor-driven (OASIS instead of W3C)
- UDDI: service registration

Examples

Older technology: Network Time.
Computers synchronise their clocks over the network.
Special Protocol (NTP) required, used since before 1985.

Modern: Web Services provide one protocol for many services.

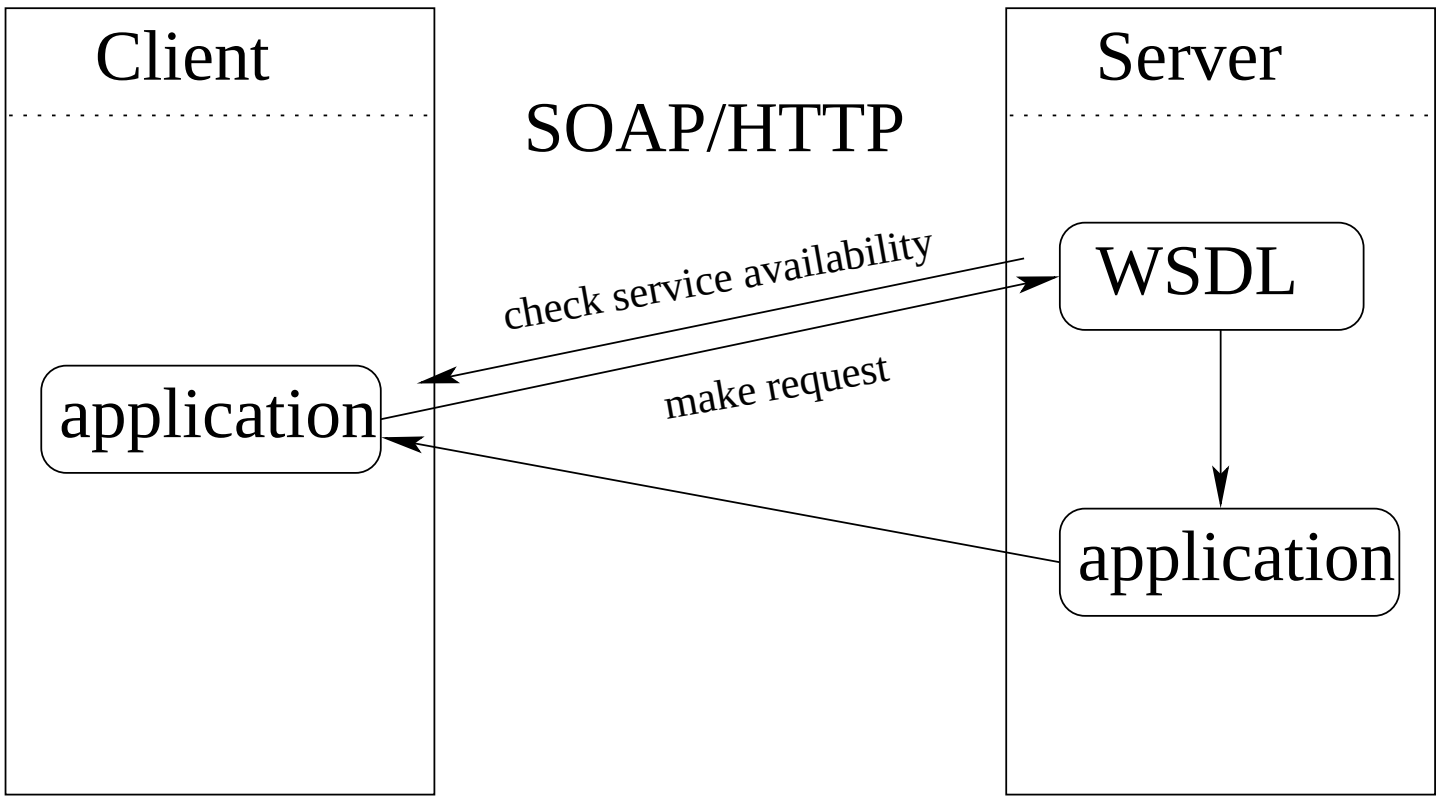
Examples:

- search engine APIs
- postal code look up
- Whois database
- Checking sport results, stock quotes etc

Technologies

- XML messages POSTed using SOAP
- Web Services Description Language (WSDL): machine-readable service description
- XML-RPC (older, simpler alternative to SOAP)
- REST (alternative to SOAP)

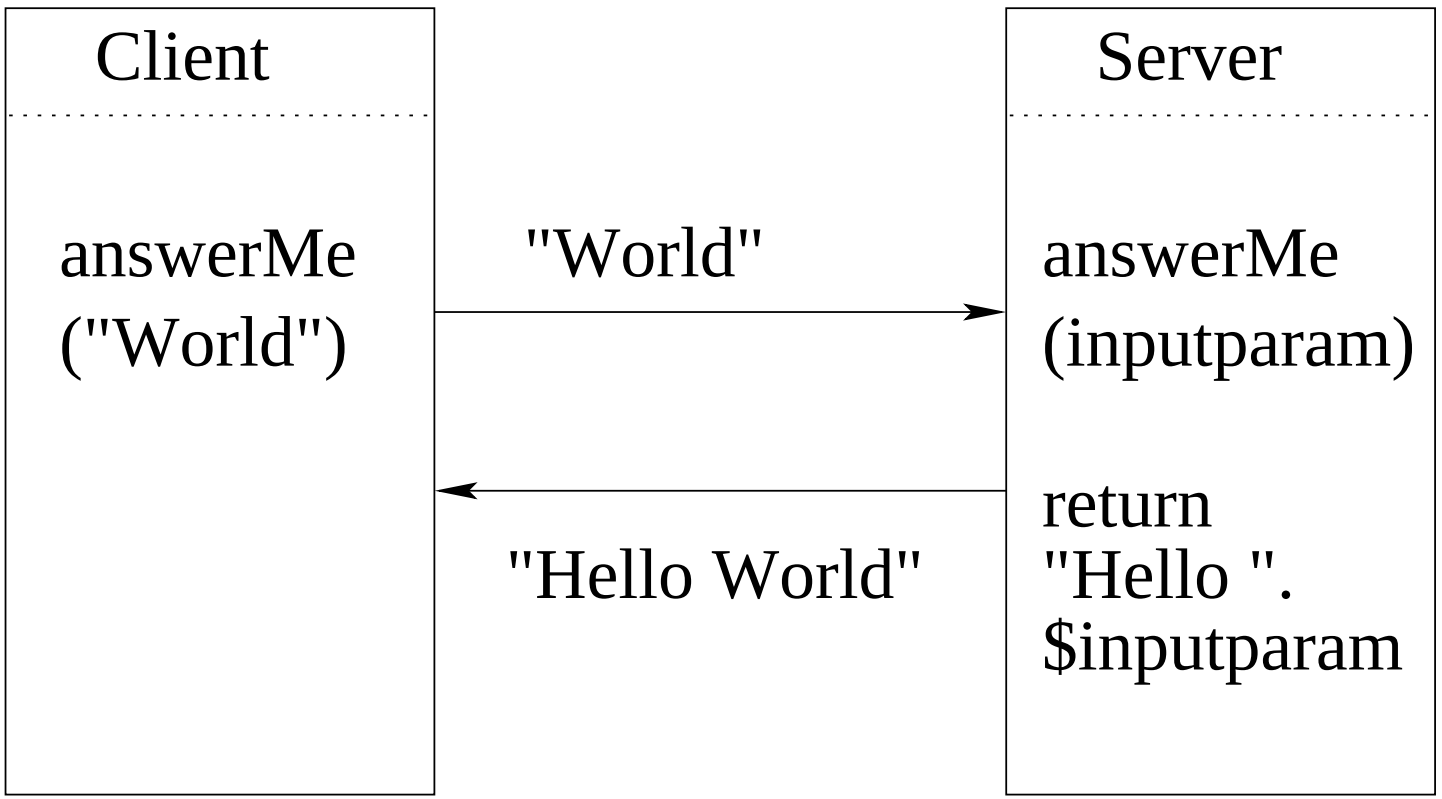
Architecture



Criticism

- complex to use, vendor-driven
- multiple standards and approaches
- each service requires custom-designed clients
- automatically created clients can be brittle

A PHP example



This simple method call ...

... requires a substantial amount of code:

- a PHP file for the service
- a file for the client
- a WSDL file

A PHP service

```
<?php
class WorldService {
      function answerMe($inputparam) {
           return "Hello ". $inputparam;
      }
}
ini_set("soap.wsdl_cache_enabled", "0");
$server = new SoapServer("helloworld.wsdl");
$server->setClass("WorldService");
$server->handle();
?>
```

A (non-WSDL) PHP client

```
<?php
$client = new SoapClient(NULL, array(
      "location" => "http:// ... /helloserver.php",
      "uri" => "something",
      "style" => SOAP_RPC,
      "use" => SOAP_ENCODED
      ));
print($client->__soapCall("answerMe",
      array(new SoapParam("World","inputparam"))));
?>
```

A (WSDL) PHP client

```
<?php
ini_set("soap.wsdl_cache_enabled", "0");
try {
      $client = new SoapClient("helloworld.wsdl");
      $result = $client->answerMe("World");
      print($result);
} catch (SoapFault $exception) {
      echo $exception;
}
?>
```

WSDL: input/output as messages

```
<?xml version ='1.0' encoding ='UTF-8' ?>
<definitions name='HelloWorld' ....
<message name='getHelloWorldRequest'>
      <part name='inputparam' type='xsd:string'/>
</message>
<message name='getHelloWorldResponse'>
      <part name='Result' type='xsd:string'/>
</message>
```

WSDL: methods as operations

```
<portType name='HelloWorldPortType'>
<operation name='answerMe'>
      <input message='tns:getHelloWorldRequest'/>
      <output message='tns:getHelloWorldResponse'/>
</operation>
</portType>
```

WSDL: binding an operation (method)

Here RPC over HTTP using SOAP

```
<binding name='HelloWorldBinding' type='tns:HelloWorldPortT
<soap:binding style='rpc'
transport='http://schemas.xmlsoap.org/soap/http'/>
<operation name='answerMe'>
...
```

WSDL: providing the server address

```
<service name='HelloWorldService'>
<port name='HelloWorldPort' binding='HelloWorldBinding'>
<soap:address location='http:// ... /helloserver.php'/>
</port>
</service>
</definitions>
```

WSDL

- Clearly, WSDL files should not be manually written.
- They can be automatically generated from the code of the service file.
- The clients can parse the WSDL file and automatically generate a list of the methods with data types offered by the service.
- Caching is used for efficiency.