*Университет за национално и световно стопанство*

*Факултет: Приложна информатика и статистика*

*Специалност: Бизнес Информатика*



**Курсов проект**

По Интернет Технологии

Тема: **Web Service Definition Language (WSDL)**

Изготвил: **Теодор Иванов Куртев**

**Фак. № 118020**

**Група 1708, Поток 177**

София, 2014г.

Съдържание

[1. Същност 2](#_Toc387162449)

[2. Бръз преглед на концепцията за уеб услуги 2](#_Toc387162450)

[3. WSDL 6](#_Toc387162451)

[a. Същност 6](#_Toc387162452)

[b. Типове (types) 7](#_Toc387162453)

[c. Съобщения (messages) 8](#_Toc387162454)

[d. Интерфейси (interfaces/protTypes) 8](#_Toc387162455)

[e. Байндинги (Bindings) 8](#_Toc387162456)

[f. Услуги (Services) 8](#_Toc387162457)

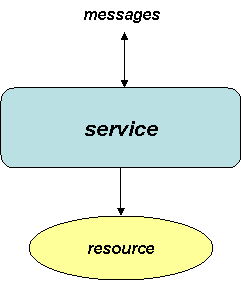
[4. История 8](#_Toc387162458)

[5. Структура 8](#_Toc387162459)

[6. Източници 8](#_Toc387162460)

1. Същност
   1. Дефиниция на W3C: WSDL представлява XML формат, който описва мрежови услуги като множество от крайни точки, които оперират чрез съобщения. Те съдържат или документно ориентирана или процедурно ориентирана информация. Операциите и съобщенията се описват абстрактно и тогава се връзват с конкретен мрежови протокол и формат на съобщението, за да се дефинира крайна точка. Свързаните конкретни крайни точки се комбинират в абстрактни крайни точки (услуги). WSDL е разширим, за да позволи описването на крайни точки и техните съобщения, без значение от формата на съобщението или мрежовите протоколи, които се използват за комуникация. Това въпреки, че единствените връзки, описани в този документ, описват как да използваме WSDL в съответствие със SOAP 1.1, HTTP GET/POST и MIME.
2. Бръз преглед на концепцията за уеб услуги

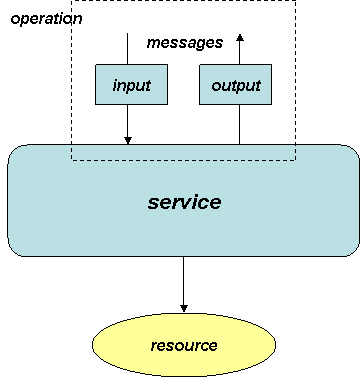
XML Web архитектура за услуги определя стандартен механизъм за предоставяне на ресурси чрез XML съобщения. Възможността да имат достъп до ресурсите чрез просто предаване на XML съобщения чрез стандартни протоколи като TCP, HTTP, SMTP значително улеснява летвата за потенциалните потребители. Терминът "Web услуга" (или просто "услуга") обикновено се отнася до част от кода, която прилага XML интерфейс към ресурсите, които в противен случай биха били трудно достъпни (вж. Фигура 1).



Тази архитектура позволява на всеки потребител с помощта на XML да се възползва от съответните Web услуги. Все пак, за да се постигне това , потребителите трябва да определят точния XML интерфейс , заедно с различни други детайли съобщение априори . XML схемата може частично да се запълни тази нужда, защото тя позволява на разработчиците да се описват структурата на XML съобщенията. XML схемата сама по себе си обаче, не може да се опише допълнителните подробности, участващи в общуването с уеб услуга.

Схемата просто казва как могат да се използват XML съобщенията, но не и как те си взаимодействат. Например, ако има един XML елемент с име Add и друг с име AddResponse, най-вероятно те ​​са свързани един с друг, но няма начин да се посочи в схемата .

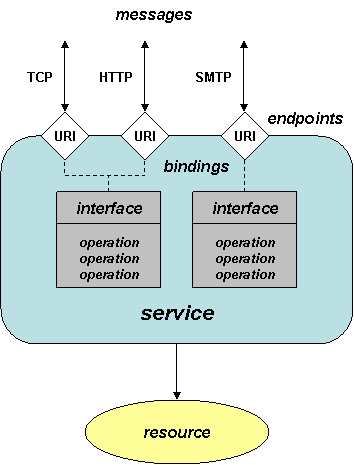
Обмена на съобщения се нарича и операция. Операциите са това, от което потребителите се интересуват най-много, тъй като те са пресечна точка на взаимодействие с услугата (вж. Фигура 2 ) . Всеки път, когато се обърне към нова уеб услуга , първо проверяваме списъка на поддържаните операции, за да се получи цялостно усещане за това, което предлага .



Обичайно е за разработчиците да групират подобни операции в интерфейси. Потребителите трябва да са наясно с тези групи , тъй като те влияят на начина, по който те пишат своя код . Това е особено важно за разработчиците , работещи с уеб услуги в обектно-ориентирани среди , тъй като XML интерфейсите могат да преобразуват програмните интерфейси (или абстрактни класове) в определен език по избор.

Потребителите трябва да знаят какъв протокол за комуникация да използват за изпращане на съобщения до услугата, заедно със специфичните механики, свързани с използването на даден протокол, като например използването на команди, хедъри и кодовете за грешки .

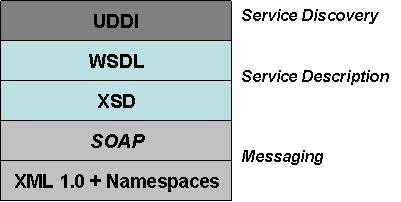
Услугите може да поддържа множество байндинги за даден интерфейс , но всеки от тях трябва да бъде достъпен в уникален адрес идентифициран с URI, или също наречен крайна точка (endpoint) (вж. Фигура 3 ) .



Потребителите трябва да открият всички детайли , описани по-горе , преди те да могат да си взаимодействат с уеб услуга. Web Services Description Language (WSDL) предоставя XML граматика за описване на тези подробности. WSDL допълва, там където XML Schema не мое да се справи чрез предоставяне на начин за групиране на съобщения в операции и операции в интерфейси. Това също осигурява начин за определяне и свързване на всеки интерфейс и комбинация от протокол заедно с endpoint . Пълна дефиниция на WSDL съдържа цялата информация, необходима, за да се позове на уеб услугата. Фирми , които искат да се направи лесно за другите да имат достъп до техните услуги следва да предоставят определения WSDL .

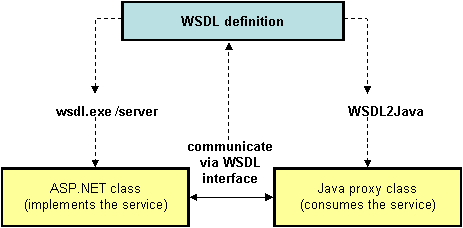
WSDL играе важна роля в цялостната архитектура на уеб услугите, тъй като той описва пълния договора за комуникация с дадено приложение (сходна с ролята на IDL в архитектурата DCOM) . Въпреки, че съществуват и други техники за описание на уеб услуги ,WS -I Basic профила Версия 1.0 препоръчва използването на WSDL и XML схема (вж. Фигура 4 ) за описание на уеб услуги . Това помага да се гарантира оперативната съвместимост на описателния слой на услугата.

.



Тъй WSDL е език четим за машините (т.к. това е просто един XML файл), инструменти и инфраструктура могат лесно да бъде изградена около него. Днес разработчиците могат да използват WSDL за да генерира код, който знае точно как да си взаимодействат с уеб услугата, който WSDL описва. Този вид подход крие досадните подробности, участващи в изпращане и получаване на SOAP съобщения по различни протоколи и прави уеб услуги, достъпни за широката публика.

Microsoft ®. NET платформата идва с помощна програма наречена wsdl.exe която генерира класове от WSDL дефиниции. Wsdl.exe може да генерира един клас за консумацията на услугата и друга за изпълнение на услугата. (Apache Axis идва с подобна програма, наречена WSDL2Java, която изпълнява същата функция за Java класове.) Класове, генерирани от същата WSDL дефиниция трябва да могат да комуникират един с друг чрез WSDL интерфейси, независимо от езиците за програмиране (виж Фигура 5).



WSDL 1.1 се счита за стандарта днес, защото се радва широка подкрепа и поддръжка. Повечето комплекти от инструменти за уеб услуги поддържат WSDL 1.1, но съществуват и някои проблеми със съвместимост в рамките на различните приложения. Много разработчици вярват, че обширната гъвкавостта на WSDL (и в резултат на сложността) е основният източник на тези проблеми. The WS-I е спомогнал да решат част от тези проблеми чрез насърчаване на разработчиците да използват някои части на спецификацията и не препоръчват използването на други.

1. WSDL
   1. Същност

WSDL e XML документ с основен елемент от XML namespace-a <http://schemas.xmlsoap.org/wsdl/> . Основния елемент може да съдържа други елементи като - types, message, interface, binding, и service. Ето и елементарен пример за структура на WSDL:

|  |
| --- |
| <!-- WSDL definition structure -->  <definitions  name="MathService"  targetNamespace="http://example.org/math/"  xmlns="http://schemas.xmlsoap.org/wsdl/"  >  <!-- abstract definitions -->  <types> ...  <message> ...  <interface> ...  <!-- concrete definitions -->  <binding> ...  <service> ...  </definition> |

В следващат таблица може да си види кратко описание на всеки един от елементите:

|  |  |
| --- | --- |
| Елемент | Описание |
| **types** | Абстрактни типове, дефинирани чрез използване на XML схемата. |
| **message** | A дефиниция на абстрактен съобщение, което може да се състои от няколко части, като всяка част може да бъде от различен тип. |
| interface | Абстрактен набор от операции, подкрепени от един или повече крайни точки (общоизвестни като интерфейс); операции се определят чрез размяна на съобщения |
| **binding** | Конкретен протокол и спецификация за формат на данни за конкретен интерфейс |
| **service** | Колекция от свързани крайни точки, където крайната точка се определя като комбинация от binding и адрес (URI) |

* 1. Типове (types)

WSDL елементът types e контейнер за определен тип XML схема дефиниции.

Можете да използвате всички конструкции на XML схемата в рамките на types елементa, дефиниране на прост тип, сложен тип, както и определенията на елементите. Следния WSDL фрагмент съдържа дефиниция на XML Schema, която определя четири елемента от тип MathInput (Add, Subtract, Multiply, и Divide) и четири елемента от тип MathOutput (AddResponse, SubtractResponse, MultiplyResponse и DivideResponse).

|  |
| --- |
| <definitions  xmlns="http://schemas.xmlsoap.org/wsdl/"  xmlns:xs="http://www.w3.org/2001/XMLSchema"  xmlns:y="http://example.org/math/"  xmlns:ns="http://example.org/math/types/"  targetNamespace="http://example.org/math/"  >  <types>  <xs:schema  targetNamespace="http://example.org/math/types/"  xmlns="http://example.org/math/types/"  >  <xs:complexType name="MathInput">  <xs:sequence>  <xs:element name="x" type="xs:double"/>  <xs:element name="y" type="xs:double"/>  </xs:sequence>  </xs:complexType>  <xs:complexType name="MathOutput">  <xs:sequence>  <xs:element name="result" type="xs:double"/>  </xs:sequence>  </xs:complexType>  <xs:element name="Add" type="MathInput"/>  <xs:element name="AddResponse" type="MathOutput"/>  <xs:element name="Subtract" type="MathInput"/>  <xs:element name="SubtractResponse" type="MathOutput"/>  <xs:element name="Multiply" type="MathInput"/>  <xs:element name="MultiplyResponse" type="MathOutput"/>  <xs:element name="Divide" type="MathInput"/>  <xs:element name="DivideResponse" type="MathOutput"/>  </xs:schema>  </types>  ...  </definitions> |

* 1. Съобщения (messages)

Messages елементът на WSDL дефинира абстрактно съобщение, което може да служи като вход или изход на операцията. Съобщенията се състоят от едни или повече родителски елемента, като всяка част е свързана с един елемент (при използване на document стил) или тип (при използване на RPC стил). Основната структура за дефиниране на съобщение е както следва (\* означава нула или повече, ? означава по избор):

|  |
| --- |
| <definitions .... >  <message name="nmtoken"> \*  <part name="nmtoken" element="qname"? type="qname"?/> \*  </message>  </definitions> |

Съобщенията и части му, трябва да бъдат именувани, което прави възможно извикването им от други места в дефиницията на WSDL. Ако дифинирате услуга в RPC стил, частите на съобщението представляват параметри на метода. Ако дифинирате услуга в documet стил, частите поросто изразяват XML елементи, които са поставени в рамките на тялото Следващият пример съдържа няколко определения съобщения, които се обръщат до елементи по име:

|  |
| --- |
| <definitions  xmlns="http://schemas.xmlsoap.org/wsdl/"  xmlns:xs="http://www.w3.org/2001/XMLSchema"  xmlns:y="http://example.org/math/"  xmlns:ns="http://example.org/math/types/"  targetNamespace="http://example.org/math/"  >  ...  <message name="AddMessage">  <part name="parameter" element="ns:Add"/>  </message>  <message name="AddResponseMessage">  <part name="parameter" element="ns:AddResponse"/>  </message>  <message name="SubtractMessage">  <part name="parameter" element="ns:Subtract"/>  </message>  <message name="SubtractResponseMessage">  <part name="parameter" element="ns:SubtractResponse"/>  </message>  ...  </definitions> |

* 1. Интерфейси (interfaces/protTypes)

portType/interface елементът във WSDL определя група от операции, известени също като интерфейс в повечето среди. За съжаление, терминът "portType" е доста объркващ, така че е по-добре да се използва термина "интерфейс". WSDL 1.2 вече е отстранява "portType" и го заменя с "интерфейс" в настоящиata версия на езика.

А interface елемент съдържа нула или повече операционни елементи. Основната структура на interface е, както следва (\* означава нула или повече):

|  |
| --- |
| <definitions .... >  <interface name="nmtoken">  <operation name="nmtoken" .... /> \*  </interdace>  </definitions> |

Всеки интерфейс трябва да има уникално име което прави възможно реферирането му от други места в WSDL дефиницията. Всеки операционен елемент съдържа комбинация от input и output елементи, и когато имаме output елемент, също може да има и fault елемент. Редът на тези елементи определя реда за обмяна на съобщения (MEP) поддържан от дадена операция.

Например, input елемент, последвано от output елемент определя request-response операция, а output елемент последвано от input елемент определя solicit-response операция. Една операция, която съдържа само един input елемент дефинира one-way операция, а една операция, която съдържа само output елемент дефинира операция notification.

Следващия пример показва интерфейса именуван **MathInterface,** който дефинира 4 операции: Add, Subtract, Multiply, и Divide

|  |
| --- |
| <definitions  xmlns="http://schemas.xmlsoap.org/wsdl/"  xmlns:xs="http://www.w3.org/2001/XMLSchema"  xmlns:y="http://example.org/math/"  xmlns:ns="http://example.org/math/types/"  targetNamespace="http://example.org/math/"  >  ...  <interface name="MathInterface">  <operation name="Add">  <input message="y:AddMessage"/>  <output message="y:AddResponseMessage"/>  </operation>  <operation name="Subtract">  <input message="y:SubtractMessage"/>  <output message="y:SubtractResponseMessage"/>  </operation>  <operation name="Multiply">  <input message="y:MultiplyMessage"/>  <output message="y:MultiplyResponseMessage"/>  </operation>  <operation name="Divide">  <input message="y:DivideMessage"/>  <output message="y:DivideResponseMessage"/>  </operation>  </interface>  ...  </definitions> |

* 1. Байндинги (Bindings)

Binding елементът във WSDL описва конкретни данни за използване на конкретен интерфейс с даден протокол. Binding елементът съдържа няколко разшираващи елементи, както и операционен WSDL елемент за всяка операция в интерфейса, който той описва. Основната структура на binding елементът е, както следва (\* означава нула или повече, ? е по избор):

|  |
| --- |
| <wsdl:definitions .... >  <wsdl:binding name="nmtoken" type="qname"> \*  <-- extensibility element providing binding details --> \*  <wsdl:operation name="nmtoken"> \*  <-- extensibility element for operation details --> \*  <wsdl:input name="nmtoken"? > ?  <-- extensibility element for body details -->  </wsdl:input>  <wsdl:output name="nmtoken"? > ?  <-- extensibility element for body details -->  </wsdl:output>  <wsdl:fault name="nmtoken"> \*  <-- extensibility element for body details -->  </wsdl:fault>  </wsdl:operation>  </wsdl:binding>  </wsdl:definitions> |

WSDL binding елемента е шаблонен. Той просто описва рамката за описване на подробностите по свързването. Действителните свързващи подробностите са предоставяни с помощта на разширяващи елементи. Тази архитектура позволява на WSDL да се развива с течение на времето, тъй като всеки елемент може да се използва в предварително определените слотове. Спецификацията на WSDL предоставя някои задължителни елементи за описване SOAP binding-и, въпреки че те са в различно пространство от имена. Следващият пример илюстрира binidng на MathInterface интерфейс към SOAP / HTTP:

|  |
| --- |
| <definitions  xmlns="http://schemas.xmlsoap.org/wsdl/"  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"  xmlns:xs="http://www.w3.org/2001/XMLSchema"  xmlns:y="http://example.org/math/"  xmlns:ns="http://example.org/math/types/"  targetNamespace="http://example.org/math/"  >  ...  <binding name="MathSoapHttpBinding" type="y:MathInterface">  <soap:binding style="document"  transport="http://schemas.xmlsoap.org/soap/http"/>  <operation name="Add">  <soap:operation  soapAction="http://example.org/math/#Add"/>  <input>  <soap:body use="literal"/>  </input>  <output>  <soap:body use="literal"/>  </output>  </operation>  ...  </binding>  ...  </definitions> |

Soap:binding елементът, указва, че това е SOAP 1.1 свързване . Той също така показва стила по подразбиране на услугата ( възможни стойности включват document или RPC ), заедно с необходимия транспортен протокол (HTTP в този случай) . Soap:operation елементът определя стойността SOAPAction HTTP хедъра за всяка операция. И soap:body определя как изглеждат частите за съобщения вътре в SOAP Body елементът (възможни стойности включват literal или encoded). Има и други свързващи специфични детайли, които могат да бъдат определени по този начин.

Използването на document стил в SOAP показва, че тялото ще съдържа XML документ , и че частите на съобщение специфицират XML елементи, които ще бъдат поставени там. Използването RPC стил в SOAP показва, че тялото ще съдържа XML представяне на извикване на метод и че частите за съобщения представляват параметри на метода.

Атрибутът use определя кодиране, което трябва да се използва за превръщане на абстрактните съобщение в конкретни такива. В случай на "encoded" , абстрактните дефиниции са преведени на конкретен формат чрез прилагане на правилата за SOAP кодиране . В случая с "literal" , дефинициите на абстрактен тип стават конкретни самите определения ( те са "literal" определения ) . В този случай , можете просто да се запознаят определенията за тип XML Schema, за да се определи формат на конкретно съобщение. Например , операцията по Add за горепосочените document/literal bindings прилича на това:

|  |
| --- |
| <SOAP-ENV:Envelope  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"  >  <SOAP-ENV:Body>  <m:Add xmlns:m="http://example.org/math/types/">  <x>3.14159265358979</x>  <y>3.14159265358979</y>  </m:Add>  </SOAP-ENV:Body>  </SOAP-ENV:Envelope> |

Забележете, че SOAP Body елемента просто съдържа инстанция на елемента Add дефинирани в схемата - това е, което прави document/literal толкова привлекателен. Сега нека да видим какво ще изглежда съобщението, като използване на RPC/encoded binding-a. Следния WSDL фрагмент съдържа ревизиран RPC/encoded binding и определенията за съобщения използват тип вместо елемент:

|  |
| --- |
| ...  <message name="AddMessage">  <part name="parameter" type="ns:MathInput"/>  </message>  <message name="AddResponseMessage">  <part name="parameter" type="ns:MathOutput"/>  </message>  ...  <binding name="MathSoapHttpBinding" type="y:MathInterface">  <soap:binding style="rpc"  transport="http://schemas.xmlsoap.org/soap/http"/>  <operation name="Add">  <soap:operation  soapAction="http://example.org/math/#Add"/>  <input>  <soap:body use="encoded"/>  </input>  <output>  <soap:body use="encoded"/>  </output>  </operation>  ...  </binding> |

С тези определения включени, операцията Add изглежда много по-различно, както можете да видите тук:

|  |
| --- |
| <SOAP-ENV:Envelope  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:xsd="http://www.w3.org/2001/XMLSchema"  xmlns:m0="http://example.org/math/types/"  >  <SOAP-ENV:Body>  <m:Add xmlns:m="http://example.org/math/">  <parameter xsi:type="m0:MathInput">  <x xsi:type="xsd:double">3.14159265358979</x>  <y xsi:type="xsd:double">3.14159265358979</y>  </parameter>  </m:Add>  </SOAP-ENV:Body>  </SOAP-ENV:Envelope> |

Използването буквални дефиниции е много по-чист и по-лесно за инструменти, за да се прав. Използването на правилата за кодиране е довело до значителни проблеми за оперативна съвместимост в целия комплекти от инструменти . Това също води до странни ситуации като не е в състояние да валидира съобщението теленото ниво срещу първоначалното определение схема ( тъй като това е абстрактна и не е вярно представяне на съобщението ) . За да помогне за облекчаване на объркването и да се улесни по-добра оперативна съвместимост , на WS -I забранява използването на кодировки напълно, включително SOAP кодиране , в Basic Profile 1.0 . Това означава, че само буквални определения могат да бъдат използвани, ако те е грижа за Basic профил 1.0 спазване. За повече информация по тази тема , вижте аргумент срещу SOAP Encoding .

Най-честата комбинация от атрибути SOAP стил / използване е документ / литерал (това, което аз бях в първия пример по-горе) . Това е по подразбиране в повечето комплекти от инструменти за днес и този, който идва с най-малко проблеми на оперативната съвместимост . Вторият най-често срещаната комбинация е RPC / кодиран, но сега, след като WS -I е забранила използването на " кодирани " , тя вече не е жизнеспособна опция. Единствената друга комбинация, която съществува, е RPC / буквален . Вижте RPC / буквално и свободата на избор за обсъждане на RPC / буквален и защо документ / литерал е по-добър .

В допълнение към обвързване на SOAP , WSDL спецификацията определя два други автомати : един за HTTP GET и POST и друг за MIME . Вижте примерите в спецификацията за повече подробности.

* 1. Услуги (Services)

WSDL service елемента определя набор от портове или крайни точки, които разкриват конкретен binding. Основната структура на service елемента е, както следва:

|  |
| --- |
| <definitions .... >  <service .... > \*  <port name="nmtoken" binding="qname"> \*  <-- extensibility element defines address details -->  </port>  </service>  </definitions> |

Трябва да се зададе на всеки интерфейс име и определен binding. След това, в рамките на interface елемента, можете да използвате разширяващ елемент за определяне на данните за адрес, специфични за свързване. Например, следния пример, определя услуга, наречена MathService който излага MathSoapHttpBinding на URL адреса на <http://localhost/math/math.asmx>:

|  |
| --- |
| <definitions  xmlns="http://schemas.xmlsoap.org/wsdl/"  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"  xmlns:xs="http://www.w3.org/2001/XMLSchema"  xmlns:y="http://example.org/math/"  xmlns:ns="http://example.org/math/types/"  targetNamespace="http://example.org/math/"  >  ...  <service name="MathService">  <port name="MathEndpoint" binding="y:MathSoapHttpBinding">  <soap:address  location="http://localhost/math/math.asmx"/>  </port>  </service>  </definitions> |

1. История
   1. WSDL 1.0 (септември 2000) е разработен от IBM, Microsoft и Ariba ,за да опишат своите инструменти SOAP уеб услуги. Реализацията е чрез комбиниране на два езика за описание: NASSL (Network Application Service спецификационен език) от IBM и SDL (Service Description Language) от Microsoft.
   2. WSDL 1.1, публикуван през март 2001 г., е формализирането на WSDL 1.0. Няма сериозни промени въведени между 1.0 и 1.1.
   3. WSDL 1.2 (юни 2003 г.) все още е работен проект на W3C. Според W3C: WSDL 1.2 е по-лесен и по-гъвкав от предишната версия. WSDL 1.2 прави опит за премахване на оперативно съвместими функции. WSDL 1.2 не е поддържан от голямата част от SOAP сървърни / доставчиците.
   4. WSDL 2.0 на W3C излиза юни 2007 година. WSDL 1.2 е преименувано на WSDL 2.0, тъй като има съществени различия от WSDL
2. Структура
3. Източници
   1. W3C Specification: <http://www.w3.org/TR/wsdl20/>
   2. Wikipedia (BG): <http://bg.wikipedia.org/wiki/WSDL>
   3. Wikipedia (EN): <http://en.wikipedia.org/wiki/Web_Services_Description_Language>
   4. MSDN - <http://msdn.microsoft.com>