Testen in SOA-omgevingen

Agenda

- Introductie
- Basisbegrippen
- Testen van services
- Testdata en XMLSpy
- Testsoorten
- Testtools
- Testuitvoering
- Validatie van testresultaten
- Drivers, stubs en mocks
- Samenvatting

2

Introductie

Wat is Service Oriented Architecture (SOA)

Service-oriëntatie, vertaling van *Service-Oriented Architecture* (SOA), is een architectuurmodel, geen technologie op zich. Centraal bestaat een SOA opgebouwd systeem uit servicecontracten. Hierbij is sprake van afnemers van diensten en leveranciers.

3 polteq

Introductie

Wat is Service Oriented Architecture (SOA)

- Methode om applicaties te integreren of te ontwerpen
  - Samenvoegen en hergebruiken van services
  - Bestaande services te verbinden met nieuw ontwikkelde services
- Goed inzetbaar bij het implementeren van bedrijfsprocessen
- Het lijkt een beetje op bouwen met lego
  - Uniforme koppelmogelijkheden
  - Verschillend van vorm/inhoud/etc.
  - Koppelingen met 'legacy' systemen

4 polteq

Introductie

Wat zijn services?

- Blokken herbruikbare functionaliteit
- Eigen verantwoordelijkheid
- Interface onafhankelijk van de implementatie
  - Formeel vastgelegd in een interfacecontract
- Eventueel GUI

5

Introductie

De positie van berichten binnen SOA

- Documenten met informatie
- XML
  - Gestandaardiseerd via XSD
  - SOAP
- Eventueel versleuteld (encrypted)
- Via servicebus en/of internet

6

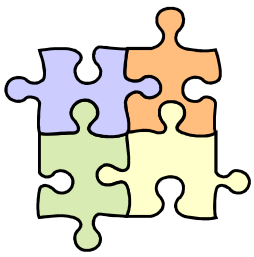
Agenda

- Introductie
- Basisbegrippen
- Testen van services
- Testdata en XMLSpy
- Testsoorten
- SoapUI
- Testuitvoering
- Validatie van testresultaten
- Drivers, stubs en mocks
- Samenvatting

7 polteq

Basisbegrippen

- XML
  - Intro
  - Voorbeeld
  - Basisuitleg
  - attributen
- XSD
  - Intro validatie
  - Voorbeeld
  - Basisuitleg
- SOAP
  - Intro communicatie
  - Voorbeeld
  - Basisuitleg
- WSDL
  - Voorbeeld
  - Basisuitleg

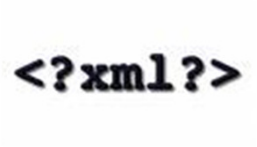


8 polteq

Basisbegrippen

eXtensible Markup Language (XML)

- Structureren van data
- Scheiden inhoud en presentatie
- Labels als bij HTML
  - Geen vaste betekenis zoals bij HTML
- Tekstformaat



9

Basisbegrippen

XML-voorbeeld

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<koppeling>
   <postcode>3821 AD</postcode>
   <huisnummer>52</huisnummer>
   <adres>
      <straat>Printerweg</straat>
      <plaats>Amersfoort</plaats>
   </adres>
</koppeling>
```

10

Basisbegrippen

XML Basis

- Documentdeclaratie

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

- Hoofdelement

```
<koppeling>
```

- Subelementen

```
<postcode>
<huisnummer>
<adres>
<straat>
<plaats>
```

11

Basisbegrippen

XML Attributen

- Meer informatie over elementen

```
<koppeling>
   <postcode>3821 AD</postcode>
   <adres straat="Printerweg" plaats="Amersfoort" />
</koppeling>
```

- Waarde van een attribuut tussen aanhalingstekens

12

Basisbegrippen

XML Validatie

- "Well Formed" XML heeft een correcte syntax:
  - Documenten hebben een hoofdelement
  - Elk element heeft een eindlabel
  - Labels zijn hoofdlettergevoelig
  - Genestheid van de labels moet kloppen
  - Attribuutwaarden staan tussen aanhalingstekens
- "Valid" XML is "Well Formed" XML gevalideerd tegen een structuur

13

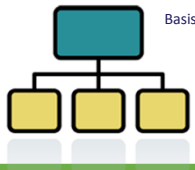
Basisbegrippen

XML Schema Definition (XSD)

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="koppeling">
      <xs:complexType>
       <xs:sequence>
              <xs:element name="postcode" type="xs:string"/>
              <xs:element name="huisnummer" type="xs:int"/>
              <xs:element name="adres" type="adresType"/>
       </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:complexType name="adresType">
       <xs:sequence>
              <xs:element name="straat" type="xs:string"/>
              <xs:element name="plaats" type="xs:string"/>
       </xs:sequence>
   </xs:complexType>
</xs:schema>
```

14

XSD Basis



- Schemadeclaratie

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

- Elementdeclaratie met complex type

```
<xs:element name="koppeling">
      <xs:complexType>
```

- Elementdeclaratie met standaard type

```
<xs:element name="postcode" type="xs:string"/>
```


Simple Object Access Protocol (SOAP)

- Communicatie protocol
- Formaat om berichten in te verzenden
- Platformonafhankelijk
- Gebaseerd op XML

- Mogelijkheid om remote procedure calls uit te voeren

Basisbegrippen

SOAP-voorbeeld

```
<soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body xmlns:m="http://www.polteq.com/AdresService">
      <m:geefAdres>
       <m:Zoekopdracht>
                <m:postcode>3821 AD</m:postcode>
                <m:huisnummer>52</m:huisnummer>
       </m:Zoekopdracht>
      </m:geefAdres>
   </soap:Body>
</soap:Envelope>
```

17

Basisbegrippen

Web Services Description Language (WSDL)

- XML gebaseerde taal
- Beschrijft webservices
  - Locatie
  - Operaties
  - Berichten

18

Basisbegrippen

WSDL-fragment

```
<message name="geefAdresSoapIn">
  <part name="parameters" element="s0:geefAdres"/>
</message>

<portType name="AdresServiceSoap">
  <operation name="geefAdres">
    <input message="s0:geefAdresSoapIn"/>
    <output message="s0:geefAdresSoapOut"/>
  </operation>
</portType>

<binding name="AdresServiceSoap" type="s0:AdresServiceSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="geefAdres">
      <soap:operation
soapAction="http://www.polteq.com/AdresService/geefAdres"
            style="document"/>
       <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
</binding>
```

19

Basisbegrippen

WSDL Basis

```
<definitions>
   <types>
      Bevat een schema met data type definities
   </types>
   <message>
      Abstracte getypeerde definitie van de data
   </message>
   <portType>
      Bevat abstracte operaties die worden ondersteund door de
      webservice
   </portType>
   <binding>
      Concrete protocol en dataspecificatie voor een portType
   </binding>
   <service>
      Bevat de concrete endpoints voor benaderen van de service
   </service>
</definitions>
```

20

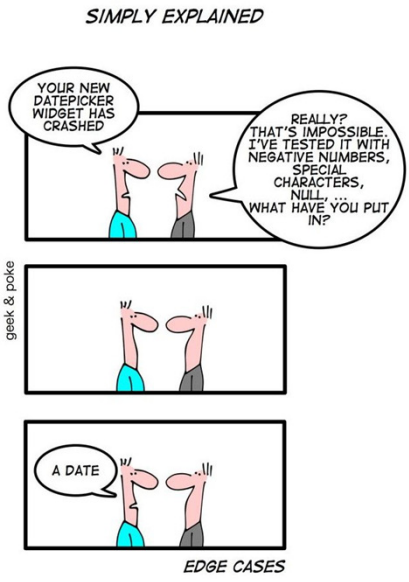
Agenda

- Introductie
- Basisbegrippen
- Testen van services
- Testdata en XMLSpy
- Testsoorten
- SoapUI
- Testuitvoering
- Validatie van testresultaten
- Drivers, stubs en mocks
- Samenvatting

21

Testen van een (web)service

- Blackbox testen via de service interface
- Gewone testtechnieken kunnen worden toegepast
- Fysieke testgevallen met XML data
- Meestal gecombineerd met een protocol zoals SOAP



22

CASUS introductie

Voor de casus gaan we een webservice testen

Wikipedia:

> Een **webservice** kan omschreven worden als een interface van een applicatiecomponent die toegankelijk is via standaard webprotocollen waarbij wordt gecommuniceerd via XML zonder menselijke tussenkomst (bijvoorbeeld SOAP)

Testen van services – deel 1

- Lees het FO
- Maak logische testgevallen
- Aanpak suggestie:
  - FO omzetten in pseudo code
  - EVT toepassen
  - Uitbreiden met grenswaarden

Agenda

- Introductie
- Basisbegrippen
- Testen van services
- Testdata en XMLSpy
- Testsoorten
- SoapUI
- Testuitvoering
- Validatie van testresultaten
- Drivers, stubs en mocks
- Samenvatting

Testbasis

- Functioneel Ontwerp zoals bij alle projecten
- Interface Agreement
  - Hierin staan de afspraken over de berichtstructuur


```
<Zoekopdracht
   xmlns="http://www.polteq.com/AdresService.asmx">
            <postcode>String</postcode>
            <huisnummer>int</huisnummer>
</Zoekopdracht>
```

Genereren van testdata

- Tools kunnen goed helpen bij het opstellen van testdata
- Structuur van berichten kan al gegenereerd worden
  - SoapUI
  - LISA
  - SOATest
  - XMLSpy
- Vervolgens zelf testgevalspecifieke data invullen
- XMLSpy kan ook al waardes invullen

27

Berichttypes

Uit de WSDL is af te lezen hoe de service communiceert

- One-way
  - portType bevat alleen input
  - Service ontvangt berichten via deze operatie
- Request-response
  - portType heeft input gevolgd door output
  - Service ontvangt een bericht en stuurt een antwoord via deze operatie
- Solicit-response
  - portType heeft output gevolgd door input
  - Service stuurt een bericht en verwacht een antwoord via deze operatie
- Notification
  - portType bevat alleen output
  - Service stuurt een bericht via deze operatie

28

Herbruikbaarheid van testdata

- Berichten zijn vaak complex
  - Voeg commentaar toe in de berichten
  - Laat voorbeeldwaardes in FO of IA opnemen

```
<!-- Testomgeving staat op test.polteq.com; acceptatie op
www.polteq.com -->
<soap:Body xmlns:m="http://www.polteq.com/AdresService">
      <m:geefAdres>
            <m:Zoekopdracht>
           <!-- Postcode bestaat uit 4 cijfers gevolgd
           door een optionele spatie en dan 2 letters -->
            <m:postcode>3821 AD</m:postcode>
            …
```

- Sla voorbeelden van berichten op!

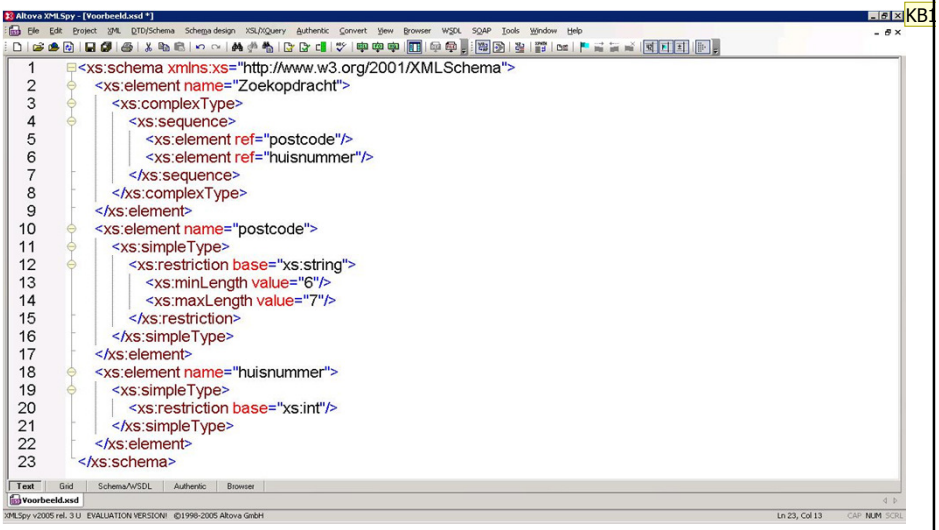
XMLSpy

- Wat kan ik met XMLSpy?
- Aan de slag met XMLSpy
- Automatisch voorbeeld genereren
- Eigen testdata valideren tegen XSD

Wat kan ik met XMLSpy?

- Veel gebruikte XML editor met ondersteuning voor
  - XML, DTD, XML Schema, XSLT, Xpath, Xquery
  - Ook SOAP, WSDL 1.1 / 2.0, Office Open XML (OOXML), XBRL
- Features zijn:
  - Sterke XML validatie, auto-aanvulling, invoer hulp, syntax kleuring, wizards, debuggers
  - En meer ondersteuning om een goed opgebouwd en valide XML bericht te creëren

XMLSpy, tekstweergave



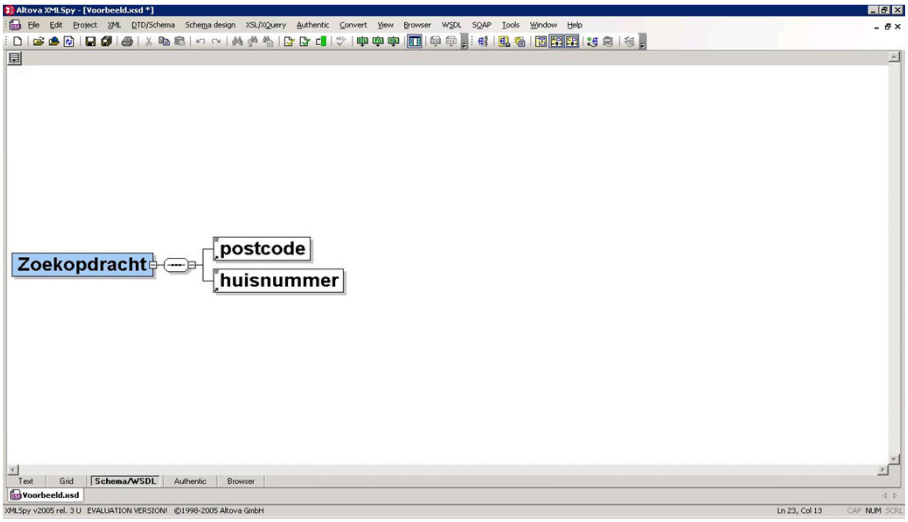

```
1  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
2    <xs:element name="Zoekopdracht">
3      <xs:complexType>
4        <xs:sequence>
5          <xs:element ref="postcode"/>
6          <xs:element ref="huisnummer"/>
7        </xs:sequence>
8      </xs:complexType>
9    </xs:element>
10   <xs:element name="postcode">
11     <xs:simpleType>
12       <xs:restriction base="xs:string">
13         <xs:minLength value="6"/>
14         <xs:maxLength value="7"/>
15       </xs:restriction>
16     </xs:simpleType>
17   </xs:element>
18   <xs:element name="huisnummer">
19     <xs:simpleType>
20       <xs:restriction base="xs:int"/>
21     </xs:simpleType>
22   </xs:element>
23 </xs:schema>
```

Dia 32

KB1 waar zitten voorbeelden van
verplicht
niet verplicht ...
Kees; 5-11-2010

Testdata en XMLSpy

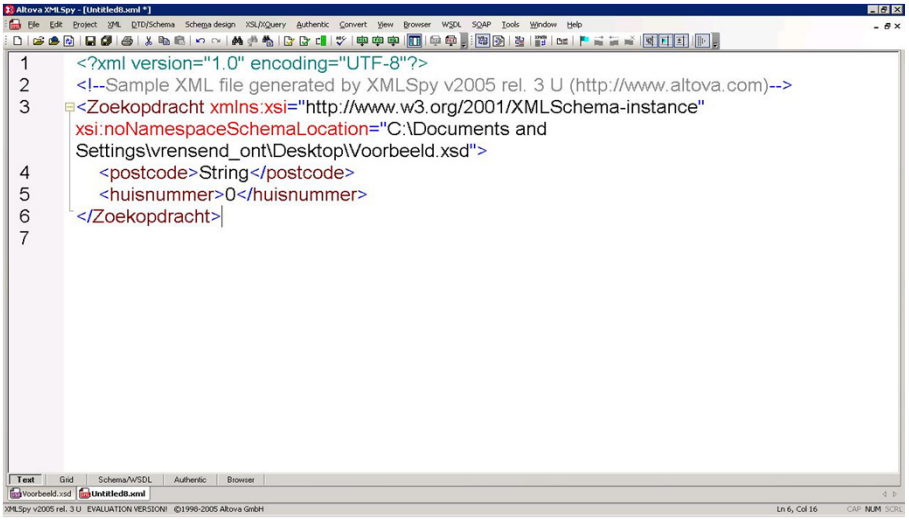
XMLSpy, grafische weergave



33

Testdata en XMLSpy

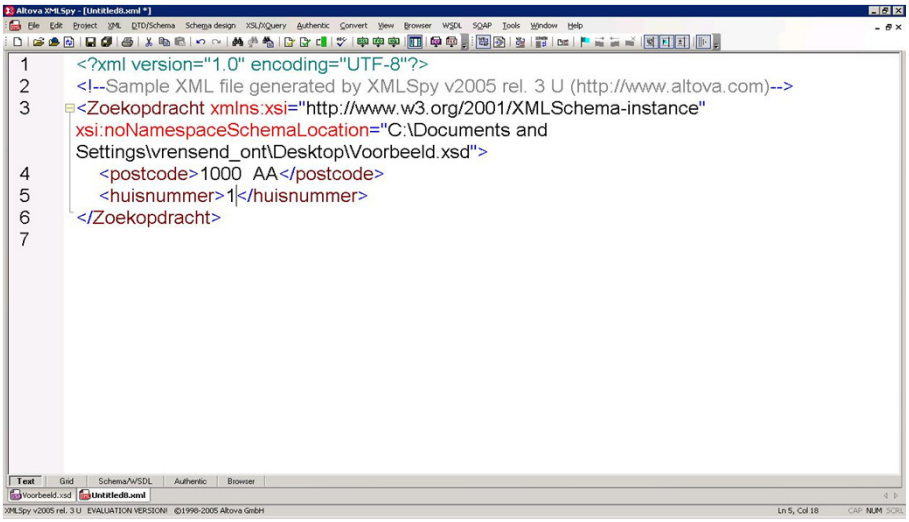
Automatisch voorbeeld genereren



34

Testdata en XMLSpy

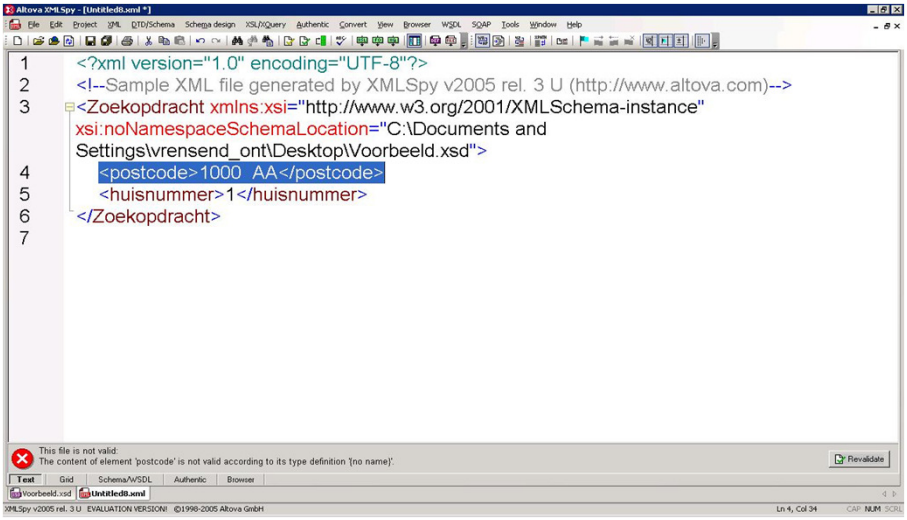
Testdata valideren tegen XSD



35

Testdata en XMLSpy

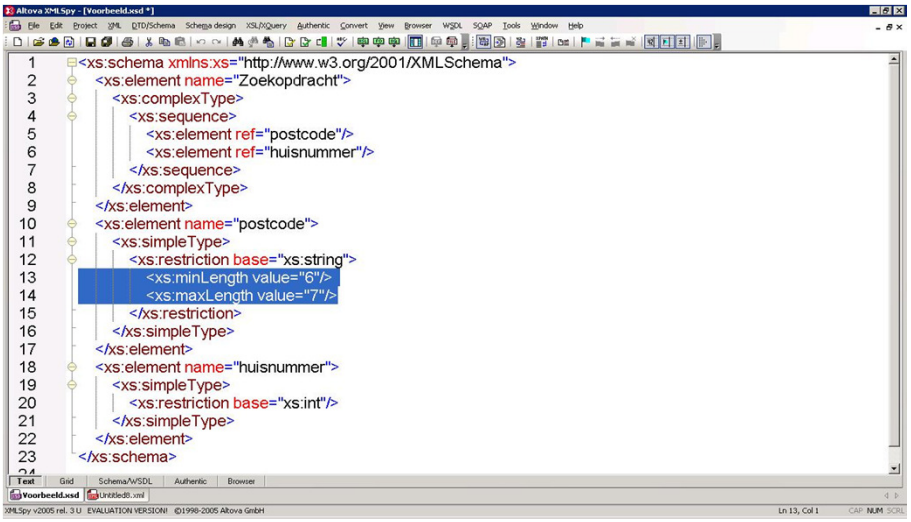
Testdata valideren tegen XSD



36

Testdata en XMLSpy

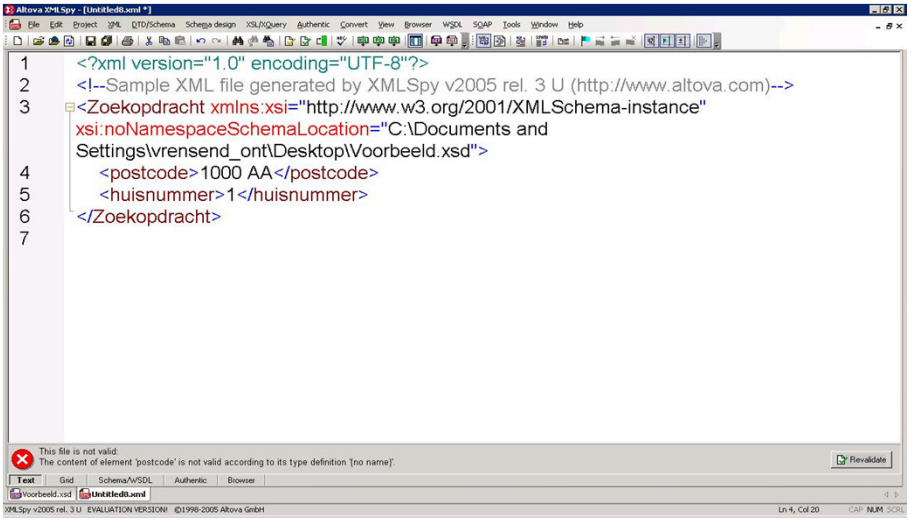
Testdata valideren tegen XSD



37

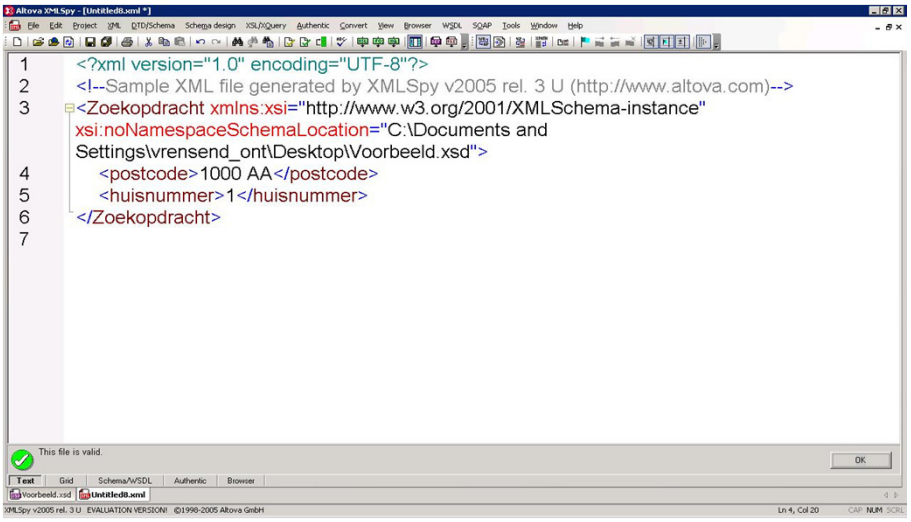
Testdata en XMLSpy

Testdata valideren tegen XSD



38

Testdata valideren tegen XSD



Testen van services – deel 2

- Zelf XML berichten maken
- Berichten valideren
- Berichten genereren

Agenda

- Introductie
- Basisbegrippen
- Testen van services
- Testdata en XMLSpy
- Testsoorten
- SoapUI
- Testuitvoering
- Validatie van testresultaten
- Drivers, stubs en mocks
- Samenvatting

41

Testsoorten in SOA omgeving

- Component test, unit test
  - Testen door de programmeur
- Service test

- Serviceintegratietest
  - Koppelingen tussen services
- Applicatietest
  - Alle services werken samen
  - Wordt het proces correct ondersteund
  - Past de applicatie in de keten
- Ketentest
  - Inbedding in technische en bedrijfsketen

42

Testsoorten

Testen van service en service interfaces

Black box test van de service via de service interface

- Bevat de interface de methoden die gevraagd worden
- Is de structuur van de berichten correct
  - Invoer
  - Uitvoer
- Is de betekenis van de responses correct
- Als dat zo is… dan werkt de service functioneel

✓ Systeemtest
✓ Acceptatietest van (web)services

43

Agenda

- Introductie
- Basisbegrippen
- Testen van services
- Testdata en XMLSpy
- Testsoorten
- SoapUI
- Testuitvoering
- Validatie van testresultaten
- Drivers, stubs en mocks
- Samenvatting

44

SoapUI

Introductie SoapUI

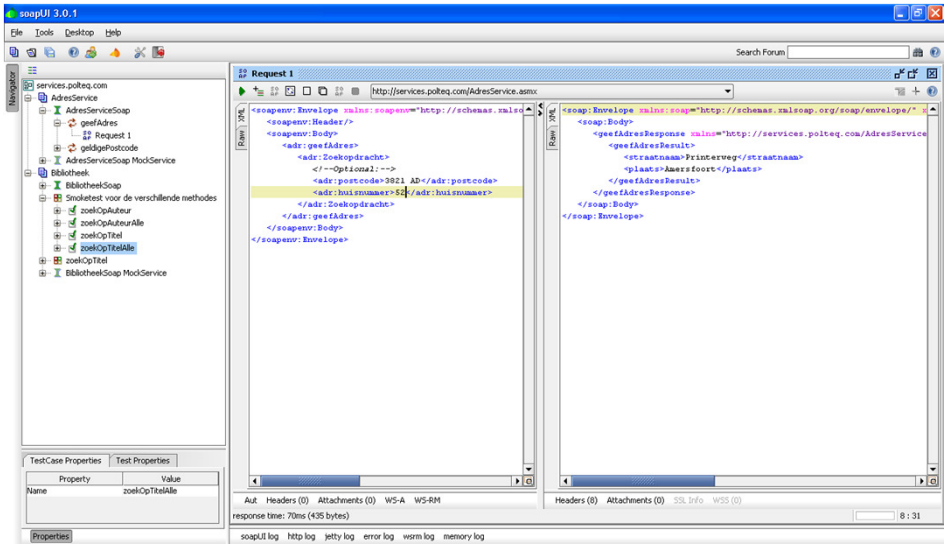
- Tool voor (voornamelijk) functioneel testen van
  - (Web)Services (SOAP, RESTfull, HTTP, JMS)
- Open source
  - Gratis versie (gebruikt in de cursus)
  - Betaalde versie met extra's
- Genereren stubs / mocks
- Mogelijkheden om uit de WSDL berichten te genereren
- Berichten via SOAP aanbieden aan de service
- Maken testsuites

45

polteq

SoapUI

Introductie SoapUI



46

polteq

Agenda

- Introductie
- Basisbegrippen
- Testen van services
- Testdata en XMLSpy
- Testsoorten
- SoapUI
- Testuitvoering
- Validatie van testresultaten
- Drivers, stubs en mocks
- Samenvatting

47

WSDL Compliancy

- Inladen van de WSDL
- Vanuit de WSDL worden de requests gegenereerd
- Als de requests gestuurd kunnen worden, dan voldoet de service in ieder geval aan de naamgeving van de methoden en de invoerberichten
- Response kan met SoapUI gevalideerd worden tegen de WSDL

48

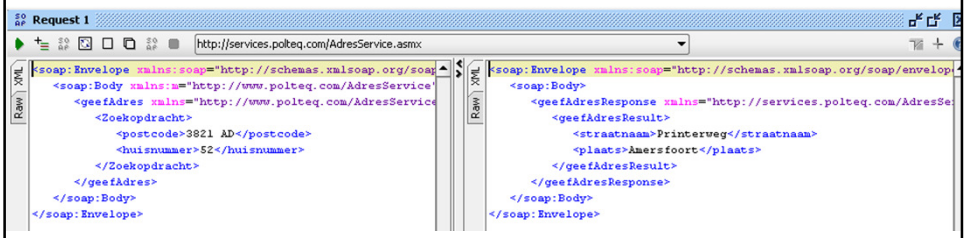
Invoer voor testgevallen klaarzetten

- Eerder gemaakte invoer voor fysieke testgevallen kan in de body van het SOAP request gezet worden
- Vervang wat SoapUI genereert

```
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body xmlns:m="http://www.polteq.com/AdresService">
      <geefAdres
xmlns="http://www.polteq.com/AdresService.asmx">
            <Zoekopdracht>
             <postcode>3821 AD</postcode>
             <huisnummer>52</huisnummer>
            </Zoekopdracht>
      </geefAdres>
   </soap:Body>
</soap:Envelope>
```

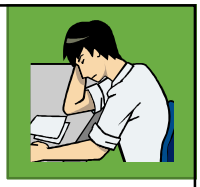

Testuitvoering met SoapUI

- Inschieten van de request



- Als er een response is, is er een technische OK
- Verwachte uitvoer vergelijken met de response

Testen van services – deel 3



- Maak project aan in SoapUI
- Intake op het testobject
- Nieuwe testsuite maken
- Testgevallen (test cases) toevoegen

51 polteq

Agenda

- Introductie
- Basisbegrippen
- Testen van services
- Testdata en XMLSpy
- Testsoorten
- SoapUI
- Testuitvoering
- Validatie van testresultaten
- Drivers, stubs en mocks
- Samenvatting

52 polteq

Functionele validatie

- Krijg je de verwachte uitvoer
  - Handmatie validatie
  - Automatische validatie
- SoapUI en validaties
  - Standaardmogelijkheden
  - Eigen validaties

Automatische validatie

Testgeval 1

- Stuur een request naar de adresService met de vraag of een postcode geldig is
- Geef als postcode 3821 AD op

Verwacht resultaat:

- Een response met
  - Geldige postcode
- Leg een lijst met geldige postcodes aan
- Laat de tool valideren of de response “Geldige postcode” bevat, wanneer een postcode uit de lijst in de request staat

Validaties met SoapUI

- Validaties worden assertions genoemd
- SoapUI biedt een aantal standaard validaties:
  - Is het een geldige SOAP Response
  - Is de response wel/geen SOAP Fault
  - Is de responsetijd korter dan gegeven aantal ms
  - Voldoet de response aan zijn XSD of WSDL
  - Bevat de response wel/niet bepaalde tekst

Eigen validaties

- Het aantal teruggegeven boeken moet kleiner of gelijk zijn aan 5

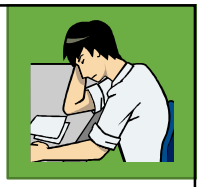
```
declare namespace
ns1='http://services.polteq.com/Bibliotheek.asmx/';
count(//ns1:Boek) <= 5
```

- Wanneer gezocht wordt op een titel, dan moet de tekst uit de zoekterm ook echt in de titel(s) voorkomen

```
declare namespace
ns1='http://services.polteq.com/Bibliotheek.asmx/';
count(//ns1:Titel[contains(lower-case(.), "cursus")]) =
count(//ns1:Boek)
```

- Voor beide als result true invullen

Testen van services – deel 4



- Assertions toevoegen aan testgevallen in SoapUI
- Stappen in testcases koppelen

57 polteq

Agenda

- Introductie
- Basisbegrippen
- Testen van services
- Testdata en XMLSpy
- Testsoorten
- SoapUI
- Testuitvoering
- Validatie van testresultaten
- Drivers, stubs en mocks
- Samenvatting

58 polteq

Drivers, stubs en mocks

- Aspecten van stubs, drivers en mocks
- Hoe maak je een stub / mock
- Zelf maken en draaien

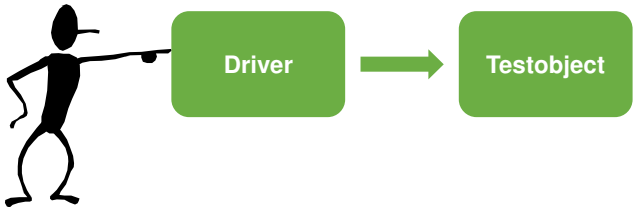
Drivers, stubs en mocks

- Drivers: stukken software die de tester in staat stellen data naar het testobject te sturen
- Stubs: stukken software die nodig zijn om nog niet bestaande stukken software te vervangen, zodat het testobject kan werken
- Mocks: stukken software vergelijkbaar met stubs, mocks leveren echter ook inzicht in hoe het testobject ze aanspreekt

Drivers, stubs en mocks

Driver

- Een driver zorgt ervoor dat er functionaliteit in het testobject kan worden aangeroepen
- Services worden vanuit applicaties aangeroepen
- Wanneer een applicatie nog niet af is, kan SoapUI worden gebruikt als driver
- Tester controleert na starten driver de uitvoer van het testobject



61

Drivers, stubs en mocks

Stub

- Stubs worden gebruikt om nog niet bestaande stukken software, die door het testobject moeten worden aangeroepen, te vervangen
- In het adresservicevoorbeeld kan, wanneer de achterliggende database met gegevens er nog niet is, een stub gebruikt worden
- Tester controleert na invoer, de uitvoer van het testobject

62

Mock

- SoapUI biedt de mogelijkheid om mockservices te genereren
- De mock kan gebruikt worden wanneer de service er nog niet is / niet af is
- Zelf in te stellen welke response teruggestuurd moet worden
- Tester controleert na invoer de uitvoer van het testobject, maar ook hoe de mock is aangesproken

Mockservice met SoapUI

- SoapUI biedt de mogelijkheid om mockservices te genereren
- Voor elke methode in de wsdl wordt een methode in de mockservice gerealiseerd
- In plaats van aanpassen request, kan hier de response aangepast worden
- Verschillende opties voor response:
  - Altijd dezelfde
  - Doorwisselende lijst
  - Random keuze uit lijst
  - Script
  - …

AdresService Mock

- Responses instelbaar (bv de verschillende adressen van Polteq-vestigingen)
  - Printerweg, Amersfoort
  - Leonard Springerlaan, Groningen
  - Singel, Dordrecht
  - Daalwijkdreef, Amsterdam
- Sequence zal ze een voor een in volgorde teruggeven
- Met script valt te programmeren welke response wanneer gegeven moet worden

polteq

Testen van services – deel 5

- Mock genereren
- Mock aanpassen
- Mock aanspreken vanuit SoapUI

polteq

Agenda

- Introductie
- Basisbegrippen
- Testen van services
- Testdata en XMLSpy
- Testsoorten
- SoapUI
- Testuitvoering
- Validatie van testresultaten
- Drivers, stubs en mocks
- Samenvatting

Samenvatting, do's en don'ts

- Service oriëntatie is een software architectuurmodel
  - Geen UI
- Communicatie met een service gaat met berichten
  - Belang van datacontracten en configuratiebeheer
  - Kennis van XML, XSD, WSDL, SOAP nodig
- Webservices, de bibliotheek casus
  - Functionele testgevallen maken met bestaande technieken
- Belang van niet-functionele aspecten
  - Testen van performance en schaalbaarheid (bijna altijd nodig)
  - SLA's nodig met leveranciers van (web)services
  - Constante regressietest nodig om continuïteit te testen
  - Beveiligingsaspecten van internet

Samenvatting, do's en don'ts

- Testdata maken met tools, zoals XMLSpy
  - Je moet kunnen 'lezen en schrijven' met XML
  - Testdata maken zonder tool feitelijk niet te doen
  - Zelfde geldt voor analyseren van XML-output
  - Validatie op XSD/WSDL nodig
- Testen van services
  - Eerst testen van de structuur van de berichten
  - Dan functioneel, black box, de service testen
  - Services integreren (ketentestaspecten)
  - Ketentesten met aanpalende software
- Testen van services doe je met tools, zoals SoapUI
  - Zonder tools nauwelijks te doen
  - Kies je tool zorgvuldig

Samenvatting, do's en don'ts

- Aspecten van testomgevingen
  - Vasthouden aan OTAP uitgangspunten
  - Goede logging en analysemogelijkheden nodig
  - Versiebeheer services, middleware, XSD's, WSDL's
- Valideren van testresultaten
  - Liefst door het tool
- Toepassing van stubs en mocks
  - De testtool is de driver
  - Testen van een service kan (haast) niet zonder stubs en mocks

- *Testen in SOA omgevingen is aantrekkelijk en veelzijdig!*

Samenvatting, do's en don'ts

Do

- Commentaar in berichten zetten
- Voorbeelden van geldige berichten opslaan
- Scheiden van testgevallen voor structuur en inhoud
- Tools inzetten

Don't

- Alles op één omgeving doen
- Aannemen dat berichten in nieuwe versies ook werken
- "Oude" technieken vergeten

polteq

polteq
Leaders in software testing

Bedankt voor jullie aandacht!

www.polteq.com