



Werner weiß...

Tim weiß...

Lotus weiß...

... Notes kann alles\*!

\*Außer Drucken. Angaben ohne Gewähr.

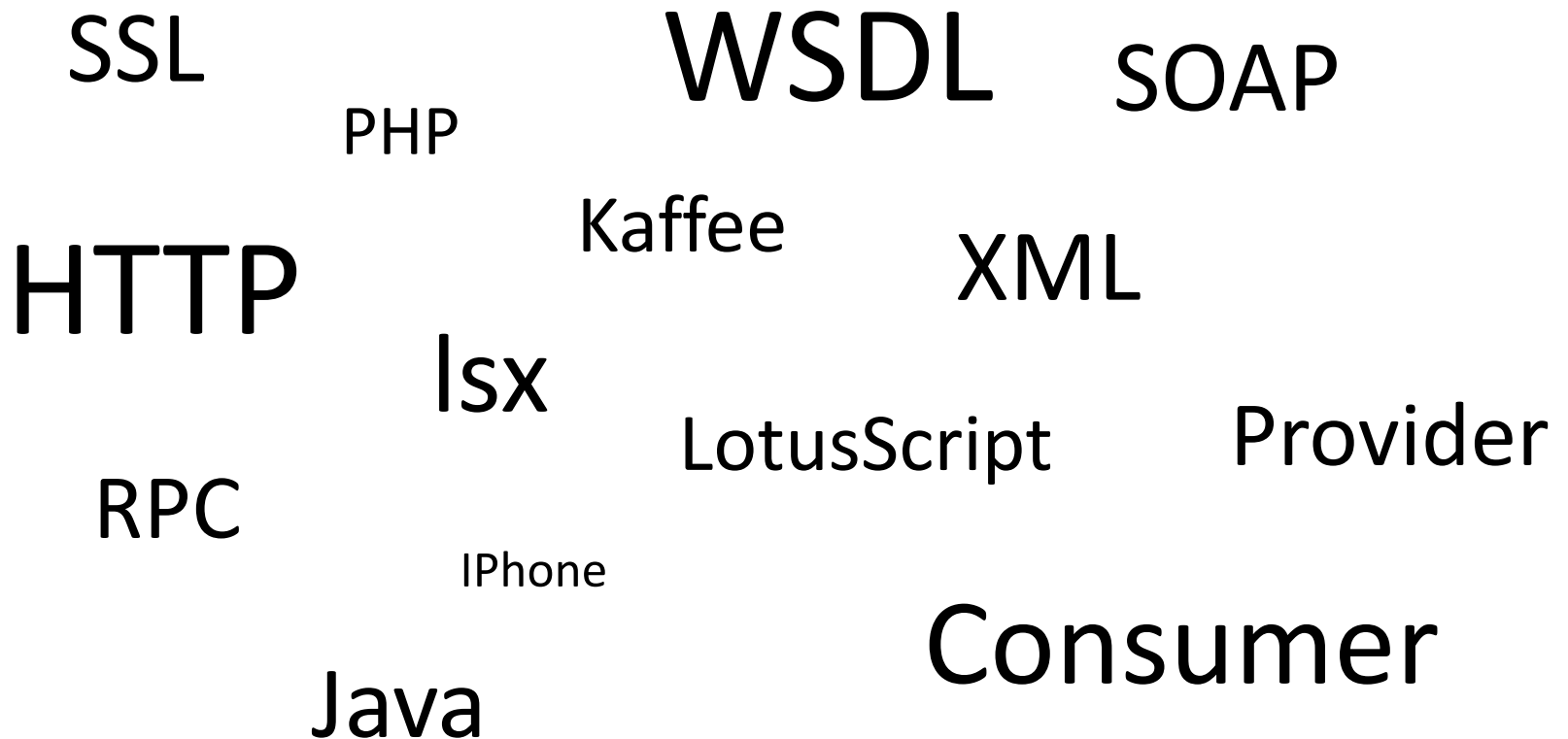
(auch Kaffee kochen)

Werner Motzet und Tim Pistor präsentieren:

# Webservices mit Lotus Notes

Werner Motzet und Tim Pistor präsentieren:

# Webservices mit Lotus Notes



A word cloud of terms related to web services and Lotus Notes. The words are arranged in a non-uniform, scattered pattern. The largest word is 'WSDL', followed by 'SOAP', 'HTTP', 'XML', 'Provider', 'Consumer', 'RPC', 'LotusScript', 'Kaffee', 'Isx', 'SSL', 'PHP', 'Java', 'iPhone', and 'Kaffee'.

SSL WSDL SOAP  
PHP  
Kaffee XML  
HTTP  
Isx LotusScript Provider  
RPC  
iPhone  
Java Consumer



Ihre Referenten

# Werner Motzet

- Idee zum Vortrag vom AdminCamp
- Bei atnotes „WernerMo“ („Wimpelnäher“)
- ProjektManager seit 1989/90
- Seit 2000 bei [isgroup] -> seit Jan.08 is@web  
<http://www.isatweb.com>
- Ursprünglich Dipl.Theol.
- 1.Sprache bayrisch <-> 2. hochdeutsch

# Tim Pistor

- Seit 2008 freier Entwickler
- Dynamische (Web-) Anwendungen unter Lotus Notes / Domino
- Co-Autor von !!HELP!!

# DEMO

## **Dynamische Webapps**

# DEMO

**Cooler Webservice**

# Unsere Ziele

# Unsere Ziele in dieser Session

- Wir durchleuchten Hintergründe und Möglichkeiten von Webservices
- Sie erlernen die ersten Schritte zum Erstellen von Webservices und...
  - ... werden in der Lage sein einen Consumer zu erstellen
  - ... werden in der Lage sein einen Provider zu erstellen

# Inhalt



# Inhalte

- Webservice Überblick
- Protokolle der Kommunikation
- Datenstrukturen
- Wer nutzt WS und wofür?
- Wie können wir WS aus Notes ansprechen?
- Wie können wir WS zur Verfügung stellen?
- Gesicherte Verbindungen
- Demos, Demos und noch mehr Demos

# Demo zum Beispiel :



# Nachbearbeitung und erste Schritte

Diese Präsentation und die Beispieldatenbanken  
finden Sie nach dem Camp auf

[www.entwicklercamp.de](http://www.entwicklercamp.de)

Was sind Webservices?

# Was sind Webservices?

- Sie ermöglichen die Zusammenarbeit von Anwendungen auf unterschiedlichen Plattformen

# Was sind Webservices?

- Sie ermöglichen die Zusammenarbeit von Anwendungen auf unterschiedlichen Plattformen
- Sie nutzen XML und Internetbasierte Protokolle (HTTP)

# Was sind Webservices?

- Sie ermöglichen die Zusammenarbeit von Anwendungen auf unterschiedlichen Plattformen
- Sie nutzen XML und internetbasierte Protokolle (HTTP)
  
- Sie halten sich an objektorientierte Programmierstandards

# Was sind Webservices?

- Sie ermöglichen die Zusammenarbeit von Anwendungen auf unterschiedlichen Plattformen
- Sie nutzen XML und internetbasierte Protokolle (HTTP)
- Sie halten sich an objektorientierte Programmierstandards
  
- Ihre Datenstrukturen und Methoden werden durch das WSDL Format beschrieben (XML)



# Was sind Webservices?

- Sie ermöglichen die Zusammenarbeit von Anwendungen auf unterschiedlichen Plattformen
- Sie nutzen XML und internetbasierte Protokolle (HTTP)
- Sie halten sich an objektorientierte Programmierstandards
- Ihre Datenstrukturen und Methoden werden durch die WSDL beschrieben.
  
- Sie sind keine Webseiten

# Was sind Webservices?

„Was für den Mensch die Webseite ist für den  
Computer der Webservice“

# DEMO

**Wetter**

# Demo – Einfacher Webservice

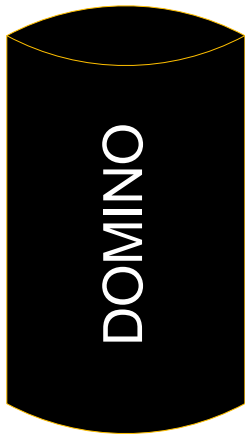
- Kontaktaufnahme zu einem Provider
- Einfacher Funktionsaufruf
- Einfacher Funktionsaufruf mit Parameter
- Kommunikation unter die Lupe genommen

# DEMO

**Einfacher Webservice**

# Demo – 1

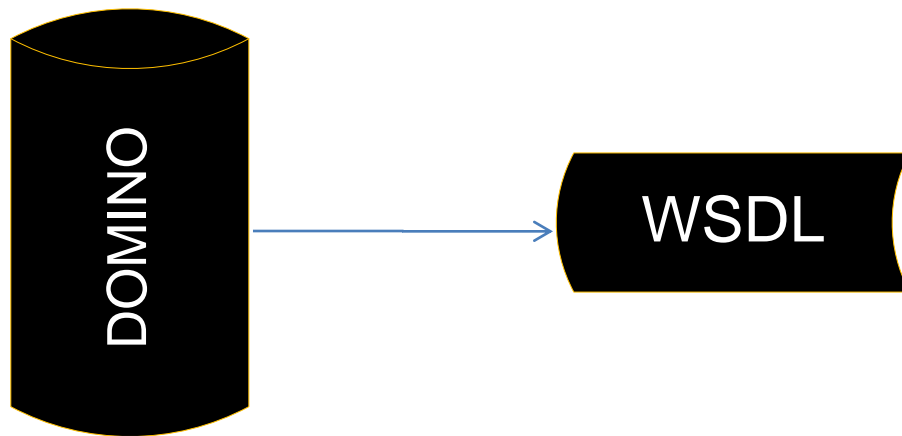
Provider



- Einfacher Funktionsaufruf
- Einfacher Funktionsaufruf mit Parameter

# Demo – 1

Provider

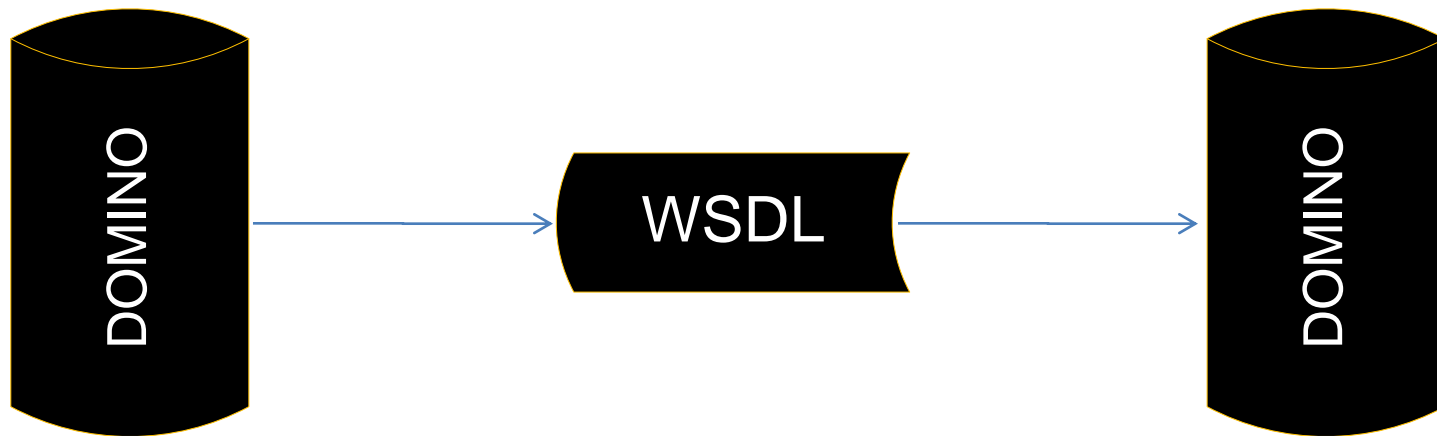


- Einfacher Funktionsaufruf
- Einfacher Funktionsaufruf mit Parameter

# Demo – 1

Provider

Consumer



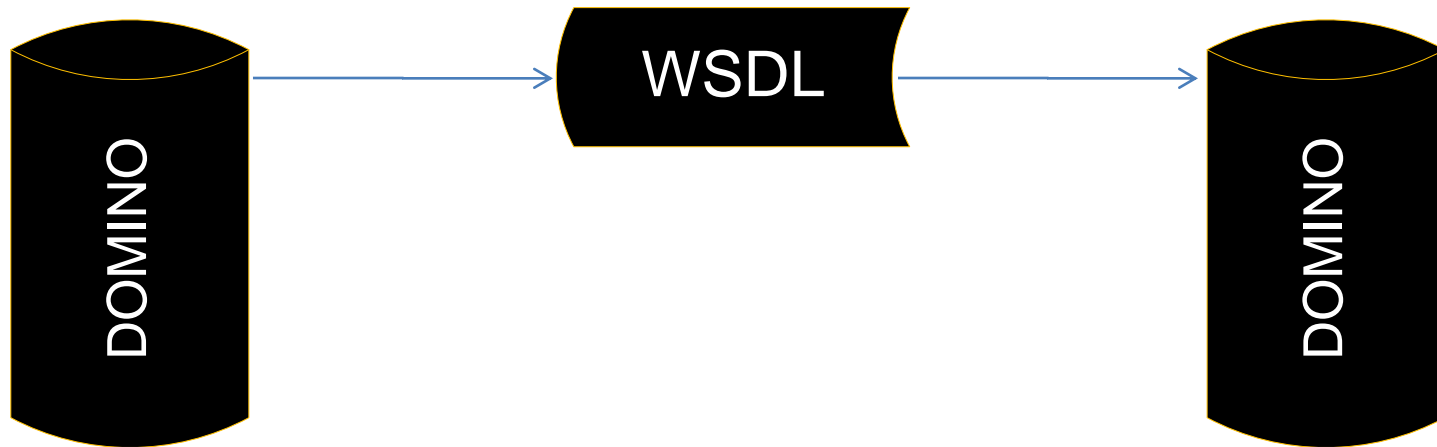
- Einfacher Funktionsaufruf
- Einfacher Funktionsaufruf mit Parameter



# Demo – 1

Provider

Consumer

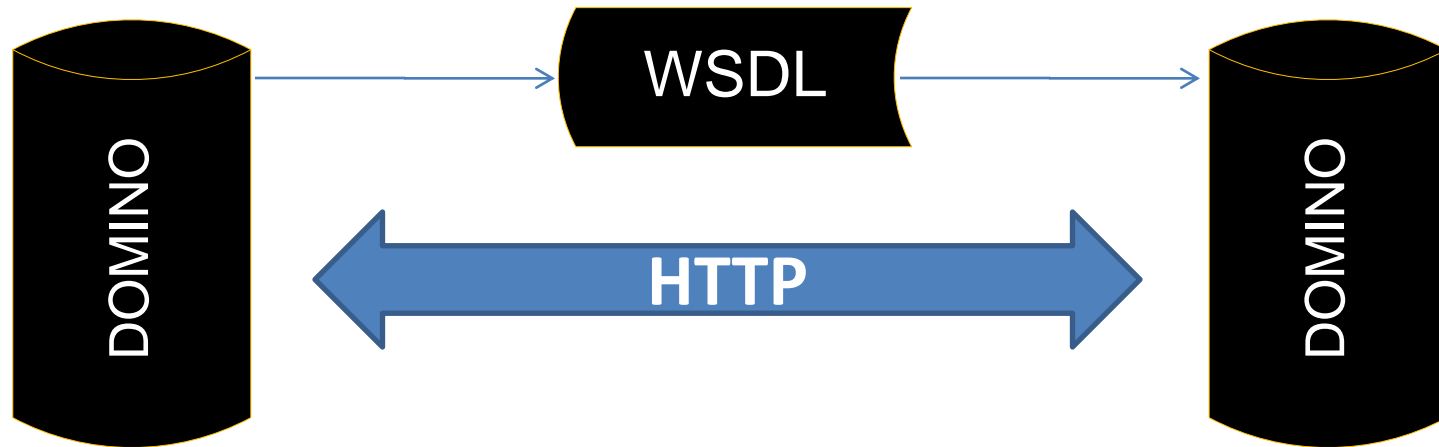


- Einfacher Funktionsaufruf
- Einfacher Funktionsaufruf mit Parameter

# Demo – 1

Provider

Consumer



- Einfacher Funktionsaufruf
- Einfacher Funktionsaufruf mit Parameter

# WSDL – Die Schnittstellenbeschreibung

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions targetNamespace="urn:DefaultNamespace" xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:apachesoap="http://xml.apache.org/xml-soap" xmlns:impl="urn:DefaultNamespace"
  xmlns:intf="urn:DefaultNamespace" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:wSDL="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wsoap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<wsdl:types>
  <schema targetNamespace="urn:DefaultNamespace" xmlns="http://www.w3.org/2001/XMLSchema">
    <element name="EINEFUNCTIONReturn" type="xsd:boolean"/>
  </schema>
</wsdl:types>
<message name="EINEFUNCTIONResponse">
  <part element="impl:EINEFUNCTIONReturn" name="EINEFUNCTIONReturn"/>
</message>
<message name="EINEFUNCTIONRequest">
</message>
<portType name="wsdlview">
  <operation name="EINEFUNCTION">
    <input message="impl:EINEFUNCTIONRequest" name="EINEFUNCTIONRequest"/>
    <output message="impl:EINEFUNCTIONResponse" name="EINEFUNCTIONResponse"/>
  </operation>
</portType>
<binding name="DominoSoapBinding" type="impl:wsdlview">
  <wsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="EINEFUNCTION">
    <wsoap:operation soapAction="EINEFUNCTION"/>
    <input name="EINEFUNCTIONRequest">
      <wsoap:body use="literal"/>
    </input>
    <output name="EINEFUNCTIONResponse">
      <wsoap:body use="literal"/>
    </output>
  </operation>
</binding>
<service name="wsdlviewService">
  <port binding="impl:DominoSoapBinding" name="Domino">
    <wsoap:address location="http://localhost"/>
  </port>
</service>
</definitions>
```

# WSDL – Die Schnittstellenbeschreibung

- Web Service Description Language (XML)
- Enthält alle nötigen Informationen über
  - Austauschprotokolle
  - Funktionen
  - Daten
  - Datentypen

# WSDL – Die Schnittstellenbeschreibung

- XML-Hauptelemente
  - Datentypen (types)
  - Nachrichten (messages)
  - Schnittstellentypen (portType)
  - Bindung (binding)
  - Ports (port)
  - Services (service)

# WSDL – Die Schnittstellenbeschreibung

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions targetNamespace="urn:DefaultNamespace" xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:apachesoap="http://xml.apache.org/xml-soap" xmlns:impl="urn:DefaultNamespace"
  xmlns:intf="urn:DefaultNamespace" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:wSDL="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wsoap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<wsdl:types>
<schema targetNamespace="urn:DefaultNamespace" xmlns="http://www.w3.org/2001/XMLSchema">
  <element name="EINEFUNCTIONReturn" type="xsd:boolean"/>
</schema>
</wsdl:types>
<message name="EINEFUNCTIONResponse">
<part element="impl:EINEFUNCTIONReturn" name="EINEFUNCTIONReturn"/>
</message>
<message name="EINEFUNCTIONRequest">
</message>
<portType name="wsdlview">
<operation name="EINEFUNCTION">
  <input message="impl:EINEFUNCTIONRequest" name="EINEFUNCTIONRequest"/>
  <output message="impl:EINEFUNCTIONResponse" name="EINEFUNCTIONResponse"/>
</operation>
</portType>
<binding name="DominoSoapBinding" type="impl:wsdlview">
<wsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
<operation name="EINEFUNCTION">
<wsoap:operation soapAction="EINEFUNCTION"/>
<input name="EINEFUNCTIONRequest">
  <wsoap:body use="literal"/>
</input>
<output name="EINEFUNCTIONResponse">
  <wsoap:body use="literal"/>
</output>
</operation>
</binding>
<service name="wsdlviewService">
<port binding="impl:DominoSoapBinding" name="Domino">
  <wsoap:address location="http://localhost"/>
</port>
```

# WSDL – Die Schnittstellenbeschreibung

- Datentypen (types)
  - Schnittstellentypen (portType)
  - Ports (port)
- Nachrichten (messages)
  - Bindung (binding)
  - Services (Service)

Definition der Datentypen, die zum Austausch der messages genutzt werden

```
<wsdl:types>  
  <schema targetNamespace="urn:DefaultNamespace"  
    xmlns="http://www.w3.org/2001/XMLSchema">  
    <element name="HELLOWORLDReturn" type="xsd:string"/>  
    <element name="HELLOWORLDNOWReturn" type="xsd:string"/>  
    <element name="PAR" type="xsd:string"/>  
    <element name="HELLOWORLDPINGPONGReturn" type="xsd:string"/>  
  </schema>  
</wsdl:types>
```

# WSDL – Die Schnittstellenbeschreibung

- Datentypen (types)
- Nachrichten (messages)
- Schnittstellentypen (portType)
- Bindung (binding)
- Ports (port)
- Services (Service)

Abstrakte Definitionen der übertragenen Daten, bestehend aus mehreren logischen Teilen, von denen jedes mit einer Definition innerhalb eines Datentyps verknüpft ist.

```
<message name="HELLOWORLDRequest">  
</message>
```

```
<message name="HELLOWORLDResponse">  
  <part element="impl:HELLOWORLDReturn" name="HELLOWORLDReturn"/>  
</message>
```



# WSDL – Die Schnittstellenbeschreibung

- Datentypen (types)
- Nachrichten (messages)
- Schnittstellentypen (portType)
- Bindung (binding)
- Ports (port)
- Services (Service)

Eine Menge von abstrakten Arbeitsschritten :

```
<portType name="exampleOne">  
  <operation name="HELLOWORLD">  
  
    <input message="impl:HELLOWORLDRequest,"  
      name="HELLOWORLDRequest"/>  
  
    <output message="impl:HELLOWORLDResponse"  
      name="HELLOWORLDResponse"/>  
  
  </operation>  
</portType>
```

# WSDL – Die Schnittstellenbeschreibung

- Datentypen (types)
- Nachrichten (messages)
- Schnittstellentypen (portType)
- **Bindung (binding)**
- Ports (port)
- Services (Service)

Bestimmt das konkrete Protokoll und Datenformat für die Arbeitsschritte

```
<binding name="DominoSoapBinding" type="impl:exampleOne">  
  <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>  
  <operation name="HELLOWORLD">  
    <wsdlsoap:operation soapAction="HELLOWORLD"/>  
    <input name="HELLOWORLDRequest">  
      <wsdlsoap:body use="literal"/>  
    </input>  
    <output name="HELLOWORLDResponse">  
      <wsdlsoap:body use="literal"/>  
    </output>  
  </operation>  
</binding>
```

(gekürzt)

# WSDL – Die Schnittstellenbeschreibung

- Datentypen (types)
- Nachrichten (messages)
- Schnittstellentypen (portType)
- Bindung (binding)
- Ports (port)
- Services (Service)

Spezifiziert eine Adresse für eine Bindung, also eine Kommunikationsschnittstelle, üblicherweise ein URI

```
<port binding="impl:DominoSoapBinding" name="Domino">  
  <wsdlsoap:address location="http://localhost"/>  
</port>
```

# WSDL – Die Schnittstellenbeschreibung

- Datentypen (types)
- Nachrichten (messages)
- Schnittstellentypen (portType)
- Bindung (binding)
- Ports (port)
- **Services (Service)**

Fassen eine Menge von verwandten Ports zusammen.

```
<service name="exampleOneService">  
  <port binding="impl:DominoSoapBinding" name="Domino">  
    <wsdlsoap:address location="http://localhost"/>  
  </port>  
</service>
```

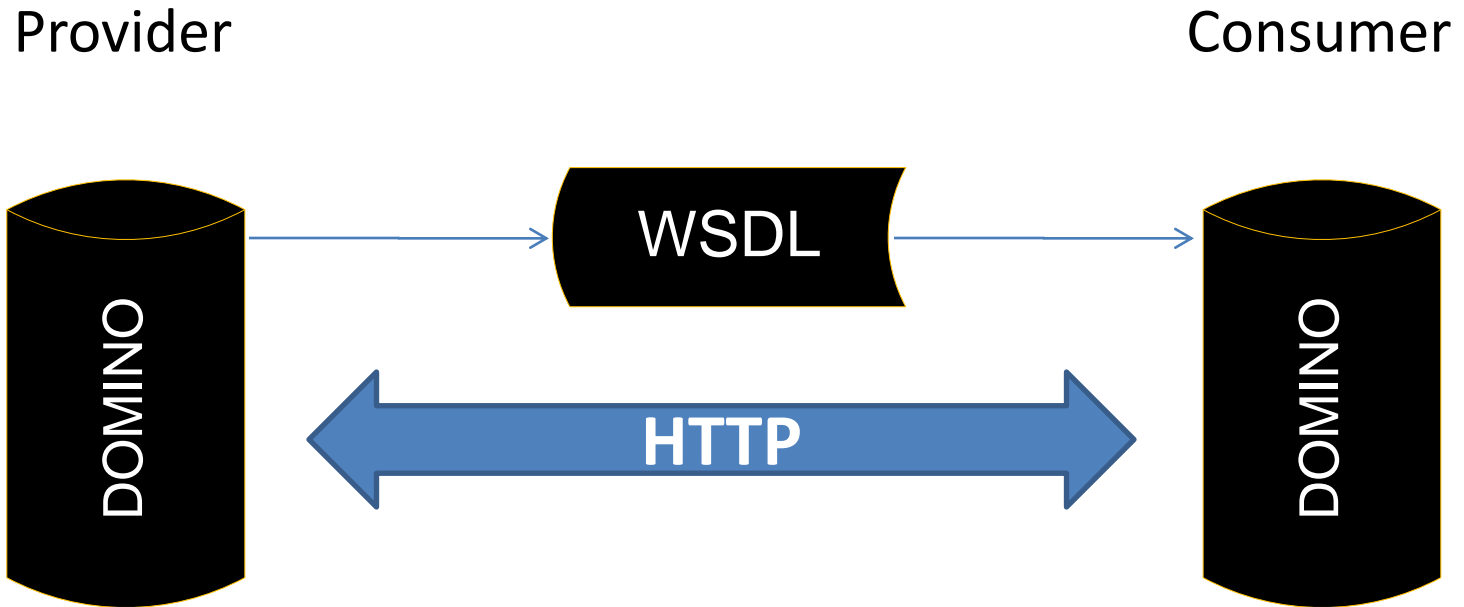
# Webservices mit Lotus Notes

SSL  
WSDL SOAP  
PHP  
Kaffee XML  
HTTP  
Isx LotusScript Provider  
RPC  
Iphone  
Java Consumer

# Datentypen

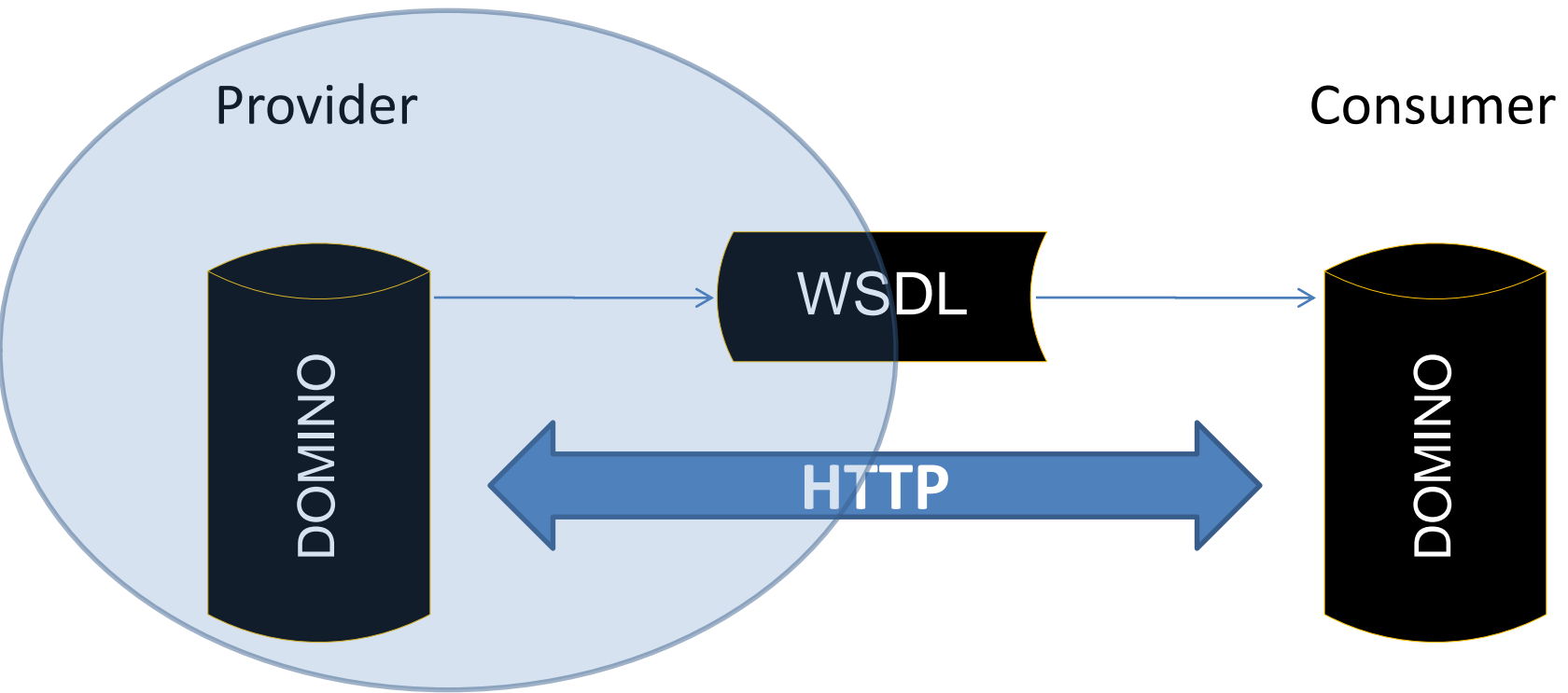
Erstellen eines Webservice Provider

# Datentypen / Erstellen eines Providers



- Einfacher Funktionsaufruf
- Einfacher Funktionsaufruf mit Parameter

# Datentypen / Erstellen eines Providers



- Einfacher Funktionsaufruf
- Einfacher Funktionsaufruf mit Parameter



# DEMO

Erstellung eines Providers

# Datentypen / Erstellen eines Providers

## 1. Neuer Webservice Provider im Designer anlegen

The screenshot displays the IBM Design Center interface. On the left, a project tree shows the structure of the 'EC 2010 Webservice Provider' project, with 'Web-Service-Provider' selected. The main area shows a table of existing providers:

Name	Alias	Geändert	Geändert von	Kommentar
Beispiel01HelloWorld		21.02.2010 19:4...	Tim Pistor/qmesh	
Beispiel02WSDLView		27.02.2010 09:5...	Tim Pistor/qmesh	

In the foreground, the 'Neuer Web-Service-Provider' dialog box is open. It contains the following fields:

- Name:** Beispiel03Demo2
- Anwendung:** EC 2010 Webservice Provider : \\comsrv01/qmesh/home\vortraege/ec10provider.nsf

Buttons for 'OK' and 'Abbrechen' are visible at the bottom of the dialog box.

# Datentypen / Erstellen eines Providers

## 2. Klasse „Beispiel03“ anlegen.

The screenshot shows the Lotus Domino Designer interface. On the left, the 'Anwendungen' (Applications) tree is expanded to 'EC 2010 Webservice Provider' > 'Code' > 'Web-Service-Provider'. Under 'Web-Service-Provider', three items are listed: 'Beispiel01HelloWorld', 'Beispiel02WSDLView', and 'Beispiel03Demo2'. The 'Beispiel03Demo2' item is selected. The main workspace is titled 'Beispiel03Demo2 (Web-Service) : (Declarations)'. It shows a 'Starten' (Start) dropdown set to 'Client' and a 'LotusScript' dropdown. The code editor contains the following LotusScript code:

```
Class Beispiel03
  Function eineFunktionMitParameter( par As Boolean) As String
    If par Then
      eineFunktionMitParameter = "ja"
    Else
      eineFunktionMitParameter = "nein"
    End If
  End Function
End Class
```

# Datentypen / Erstellen eines Providers

## 3. Provider konfigurieren

The screenshot displays the Lotus Symphony software interface for configuring a Web-Service Provider. The left-hand navigation pane shows the project structure under 'EC 2010 Webservice Provider', with 'Web-Service-Provider' expanded to show 'Beispiel03Demo2'. The central pane shows the 'Objekte' view for 'Beispiel03Demo2 (Web-Service)', listing '(Options)', '(Declarations)', 'Initialize', and 'Terminate'. The right-hand pane shows the configuration for 'Beispiel03Demo2 (Web-Service) : (Declarations)'. The 'Starten' dropdown is set to 'Client' and the 'LotusScript' dropdown is set to 'LotusScript'. The code editor shows the following LotusScript code:

```
Class Beispiel03
  Function eineFunktionMitParameter( par As Boolean) As String
    If par Then
      eineFunktionMitParameter = "ja"
    Else
      eineFunktionMitParameter = "nein"
    End If
  End Function
End Class
```

The configuration panel below the code editor shows the following fields:

- Name: Beispiel03Demo2
- Alias: (empty)
- Kommentar: (empty)
- Optionen:  Wamen, wenn WSDL-Schnittstelle modifiziert wird
- Port Type: Beispiel03
- Klasse: (empty)

# Datentypen / Erstellen eines Providers

## 3. Provider konfigurieren

The screenshot displays the Lotus Designer interface for configuring a Web-Service Provider. The left-hand pane shows the project structure under 'EC 2010 Webservice Provider', with 'Web-Service-Provider' expanded to show 'Beispiel03Demo2'. The central pane shows the object tree for 'Beispiel03Demo2 (Web-Servi)' with sub-items: '(Options)', '(Declarations)', 'Initialize', and 'Terminate'. The right-hand pane shows the configuration for 'Beispiel03Demo2 (Web-Service) : (Declarations)'. The configuration panel includes a 'Web-Service' dropdown menu and several options:

- Als Webbenutzer ausführen
- Ausführen im Namen von \_\_\_\_\_
- Remote-Debugging zulassen
- Profiling für diesen Web-Service aktivieren
- Laufzeit-Sicherheitsstufe festlegen: (1 = am sichersten)
- 1. Beschränkte Operationen nicht zulassen
- Vorgabezugriff für diesen Web-Service
- Alle Leser und höhere
- OtherDomainServers: comsrv01/qmesh, Tim Pistor/qmesh
- Benutzer mit öffentl. Zugriff dürfen diesen Web-Service nutzen

# Datentypen / Erstellen eines Providers

## 3. Provider konfigurieren

The screenshot displays the Lotus Domino Designer interface for configuring a Web-Service Provider. The left sidebar shows the project structure under 'EC 2010 Webservice Provider', with 'Web-Service-Provider' expanded to show 'Beispiel03Demo2'. The main workspace is titled 'Beispiel03Demo2 (Web-Service) : (Declarations)'. It features a toolbar with 'WSDL importieren', 'WSDL exportieren', 'WSDL anzeigen', and 'Print'. Below the toolbar, there are tabs for 'Objekte' and 'Referenz', and a 'Starten' dropdown set to 'Client'. The 'LotusScript' editor shows the following code:

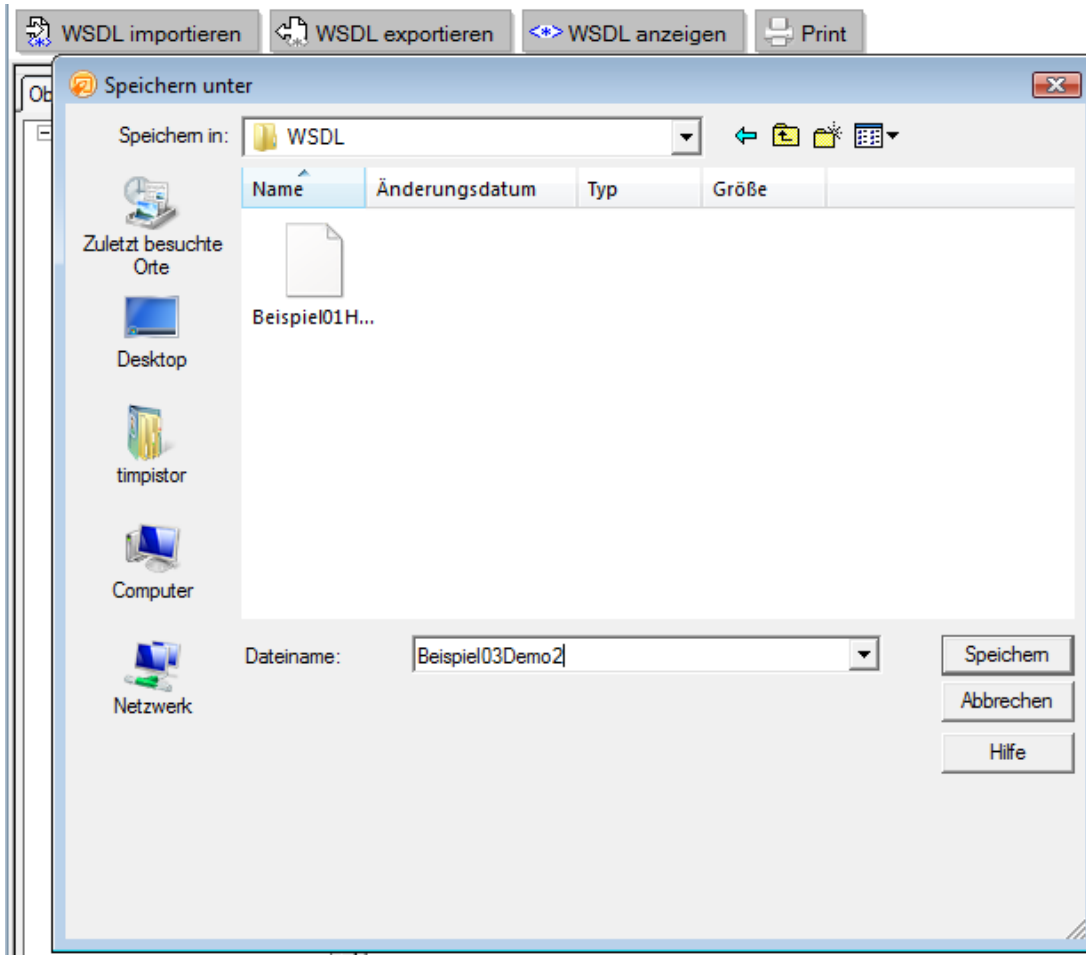
```
Class Beispiel03
  Function eineFunktionMitParameter( par As Boolean) As String
    If par Then
      eineFunktionMitParameter = "ja"
    Else
      eineFunk
    End If
  End Function
End Class
```

An inset window titled 'Web-Service' is open, showing configuration options:

- Optionen
- Programmiermodell:  RPC  Nachricht
- SOAP-Nachrichtenformat: Doc/literal
- Operationsname in SOAP-Aktion aufnehmen
- Port Type-Name: Beispiel03
- Service-Element-Name: Beispiel03Service
- Service-Port-Name: Domino

# Datentypen / Erstellen eines Providers

## 4. WSDL Exportieren



# Datentypen / Erstellen eines Providers

- Es sind alle „Standard“ Datentypen erlaubt.
  - String
  - Integer
  - Double
  - Float
  - Boolean
  - Date



# Datentypen / Erstellen eines Providers

- Es sind alle „Standard“ Datentypen erlaubt.
  - `Public a1 As Boolean`
    - `<element name="A1" type="xsd:boolean"/>`
  - `Public a2 As Integer`
    - `<element name="A2" type="xsd:short"/>`
  - `Public a3 As Double`
    - `<element name="A3" type="xsd:double"/>`
  - `Public a4 As String`
    - `<element name="A4" type="xsd:string"/>`

# Datentypen / Erstellen eines Providers

- Es sind alle „Standard“ Datentypen erlaubt.
  - Public a5 As Long
    - `<element name="A5" type="xsd:int"/>`
  - Public a6 As Byte
    - `<element name="A6" type="xsd:unsignedByte"/>`
  - Public a7 As Single
    - `<element name="A7" type="xsd:float"/>`

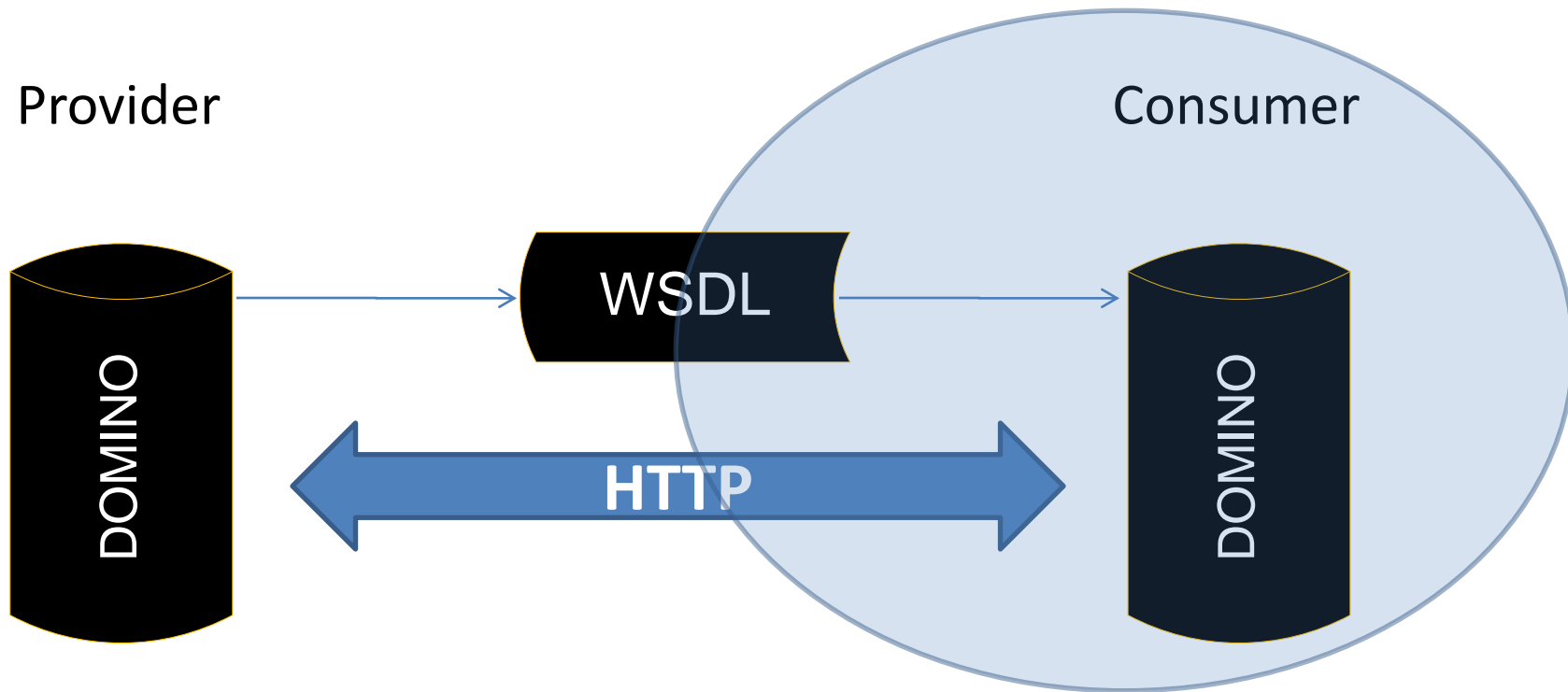
# Datentypen / Erstellen eines Providers

1. Class Beispiel04
2.     Function getDatentypen() As Datentypen
3.         Set getDatentypen = New Datentypen
4.     End Function
5. End Class
  
6. Class Datentypen
7.     Public a1 As Boolean
8.     Public a2 As Integer
9.     Public a3 As Double
10.    Public a4 As String
11.    Public a5 As Long
12.    Public a6 As Byte
13.    Public a7 As Single
14. End Class

# Konsumieren

Erstellen eines Webservice Consumers

# Konsumieren - Erstellen eines Consumers



-Einfacher Funktionsaufruf mit Parameter  
aus Beispiel05Datentypen

# Konsumieren - Erstellen eines Consumers

1. Class Beispiel05
2.       Function getDateTypenAsString(par As Datentypen) As String
3.               getDateTypenAsString = par.toString
4.       End Function
5. End Class
  
6. Class Datentypen
7.       Public a1 As Boolean               Public a2 As Integer
8.       Public a3 As Double               Public a4 As String
9.       Public a5 As Long               Public a6 As Byte
10.       Public a7 As Single
- 11.
12.       Public Function toString As String
13.               toString = a1 & " - " & a2 & " - " & a3& " - " & a4 & " - " & a5& " - " & a6& " - " & a7
14.       End Function
15. End Class

# Konsumieren - Erstellen eines Consumers

The screenshot displays the IBM WebSphere Administrator console interface. On the left, a navigation tree shows the 'EC 2010 Webservice Consumer' application selected. The main area shows a table of existing consumers and a dialog box for creating a new one.

Name	Sprachtyp	Geändert	Geändert von	Kommentar
Beispiel01HelloWorldConsumer	LotusScript	21.02.2010 19:4...	Tim Pistor/qmesh	

**Neuer Web-Service-Konsument**

Neuen Web-Service-Konsumenten erstellen.

Name:

Sprache:

Web-Service-Beschreibung abrufen von

Lokale WSDL-Datei

URL, die auf eine WSDL-Datei verweist

Anwendung:

# Konsumieren - Erstellen eines Consumers

The screenshot displays a software development environment with two main panes. The left pane, titled 'Objekte', shows a tree view for 'Beispiel05Datentypen (Web-Service-Konsument)'. The right pane, titled 'Beispiel05Datentypen (Web-Service-Konsument) : (Declarations)', shows the corresponding code declarations.

**Objekte** | Referenz

- Beispiel05Datentypen (Web-Service-Konsument)
  - (Options)
  - (Declarations)
  - Initialize
  - Terminate

**Beispiel05Datentypen (Web-Service-Konsument) : (Declarations)**

```
%INCLUDE "Isxsd.Iss"  
Class DATENTYPEN As XSD_ANYTYPE  
  
    Public A1 As Boolean  
    Public A2 As Integer  
    Public A3 As Double  
    Public A4 As String  
    Public A5 As Long  
    Public A6 As Byte  
    Public A7 As Single  
  
    Sub NEW  
    End Sub  
  
End Class  
  
Class Beispiel05 As PortTypeBase  
  
    Sub NEW  
        Call Service.Initialize ("UmDefaultNamespaceBeispiel05Service", _  
            "Beispiel05Service.Domino", "http://localhost", _  
            "Beispiel05")  
  
    End Sub  
  
    Function GETDATENTYPENASSTRING(PAR As DATENTYPEN) As String  
        Let GETDATENTYPENASSTRING = Service.Invoke("GETDATENTYPENASSTRING", PAR)  
    End Function  
  
End Class
```



# Konsumieren - Erstellen eines Consumers

```
Objekte Referenz
Beispiel05Datentypen (Web-Service-Konsument)
  (Options)
  (Declarations)
  Initialize
  Terminate

Beispiel05Datentypen (Web-Service-Konsument) : (Declarations)

%INCLUDE "Isxsd.Iss"
Class DATENTYPEN As XSD_ANYTYPE

  Public A1 As Boolean
  Public A2 As Integer
  Public A3 As Double
  Public A4 As String
  Public A5 As Long
  Public A6 As Byte
  Public A7 As Single

  Sub NEW
  End Sub

End Class

Class Beispiel05 As PortTypeBase

  Sub NEW
    Call Service.Initialize ("UmDefaultNamespaceBeispiel05Service", _
      "Beispiel05Service.Domino", "http://localhost", _
      "Beispiel05")
  End Sub

  Function GETDATENTYPENASSTRING(PAR As DATENTYPEN) As String
    Let GETDATENTYPENASSTRING = Service.Invoke("GETDATENTYPENASSTRING", PAR)
  End Function

End Class
```

# Konsumieren - Erstellen eines Consumers

Objekte Referenz

Beispiel05Datentypen (Web-Service-Konsument)

- (Options)
- Declarations
- Initialize
- Terminate

Beispiel05Datentypen (Web-Service-Konsument) : (Declarations)

```
%!INCLUDE "Isxsd.iss"  
Class DATENTYPEN As XSD_ANYTYPE  
  
Public A1 As Boolean  
Public A2 As Integer  
Public A3 As Double  
Public A4 As String  
Public A5 As Long  
Public A6 As Byte  
Public A7 As Single  
  
Sub NEW  
End Sub  
  
End Class  
  
Class Beispiel05 As PortTypeBase  
  
Sub NEW  
Call Service.Initialize ("UrnDefaultNamespaceBeispiel05Service", _  
"Beispiel05Service.Domino", "http://192.168.1.11/home/vortraege/ec10provider.nsf/Beispiel05Datentypen", _  
"")  
End Sub  
  
Function GETDATENTYPENASSTRING(PAR As DATENTYPEN) As String  
Let GETDATENTYPENASSTRING = Service.Invoke("GETDATENTYPENASSTRING", PAR)  
End Function  
  
End Class
```

# Konsumieren - Erstellen eines Consumers

Sub NEW

```
Call Service.Initialize ("UrnDefaultNamespaceBeispiel05Service", _  
"Beispiel05Service.Domino", "http://192.168.1.11/home/vortraege/ec10provider.nsf/Beispiel05Datentypen", _  
")
```

End Sub

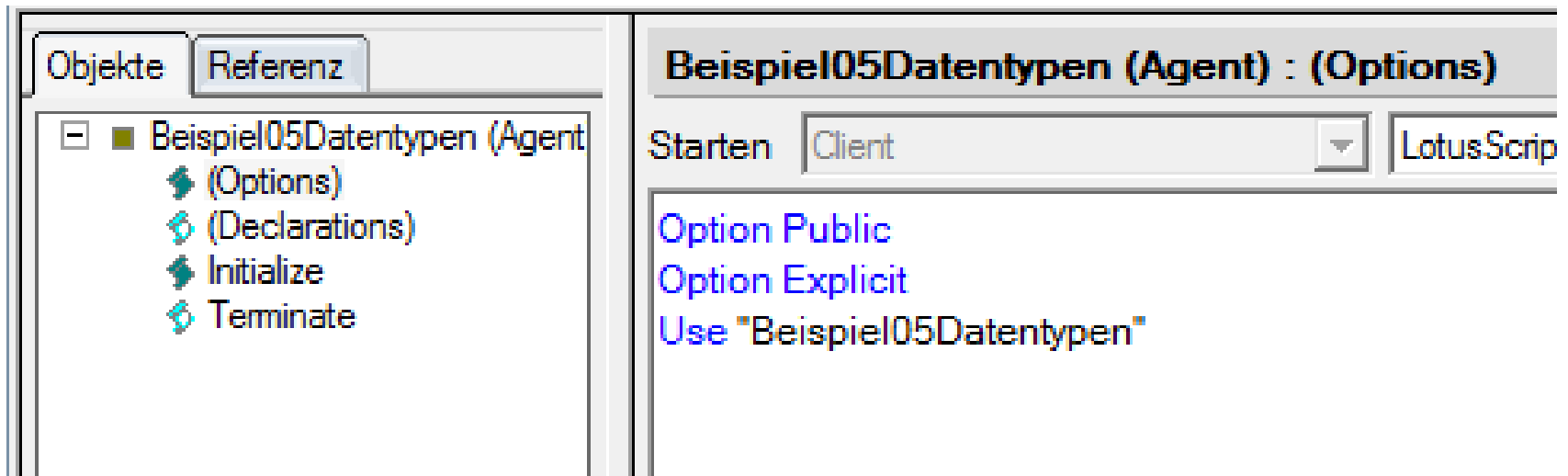
- URL zur Datenbank + „/“ Webservicename

# Konsumieren - Erstellen eines Consumers

Der fertige Consumer kann nun in Agenten, Buttons usw. eingebunden werden.

# Konsumieren - Erstellen eines Consumers

Der fertige Consumer kann nun in Agenten, Buttons usw. eingebunden werden.



# Konsumieren - Erstellen eines Consumers

The screenshot displays a software development interface with two main panels. The left panel, titled 'Objekte' and 'Referenz', shows a tree view for the agent 'Beispiel05Datentypen (Agent)'. The tree includes the following items:

- (Options)
- (Declarations)
- Initialize
- Terminate

The right panel, titled 'Beispiel05Datentypen (Agent) : Initialize', shows the code for the 'Initialize' subprocedure. The code is written in LotusScript and includes the following lines:

```
Sub Initialize
  Dim data As New Datentypen
  Dim bsp5 As New Beispiel05

  data.a1 = False
  data.a2 = 2
  data.a3 = 3.3333
  data.a4 = "vier"
  data.a5 = 5.5555
  data.a6 = 6
  data.a7 = 7

  MsgBox bsp5.getDatentypenAsString (data)
End Sub
```

At the top of the right panel, there is a 'Starten' button set to 'Client' and a 'LotusScript' dropdown menu.

# Konsumieren - Erstellen eines Consumers

Ein Webservice Consumer in PHP erstellt :

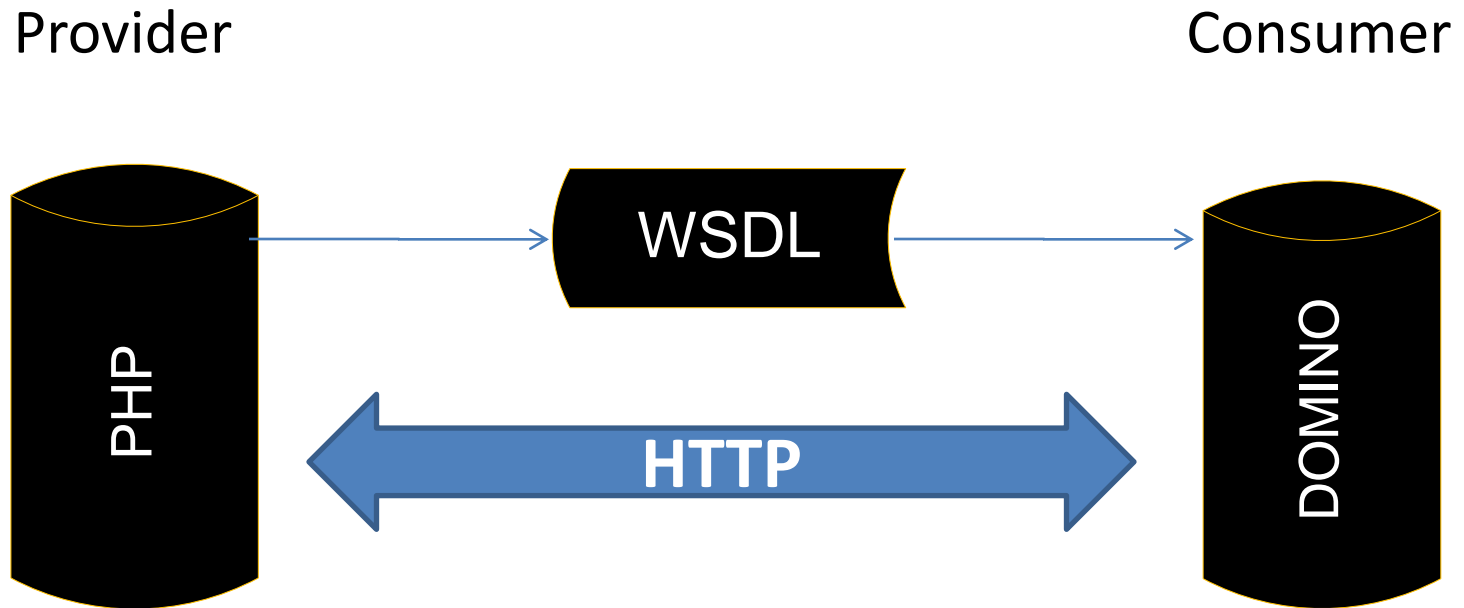
1. `$bsp1url = "http://server/dbnsf/provider?WSDL";`
2. `$bsp1client = new SoapClient($bsp1url);`
3. `$res1 = $bsp1client->helloworld();`

# Kommunikation

Datenaustausch per XML

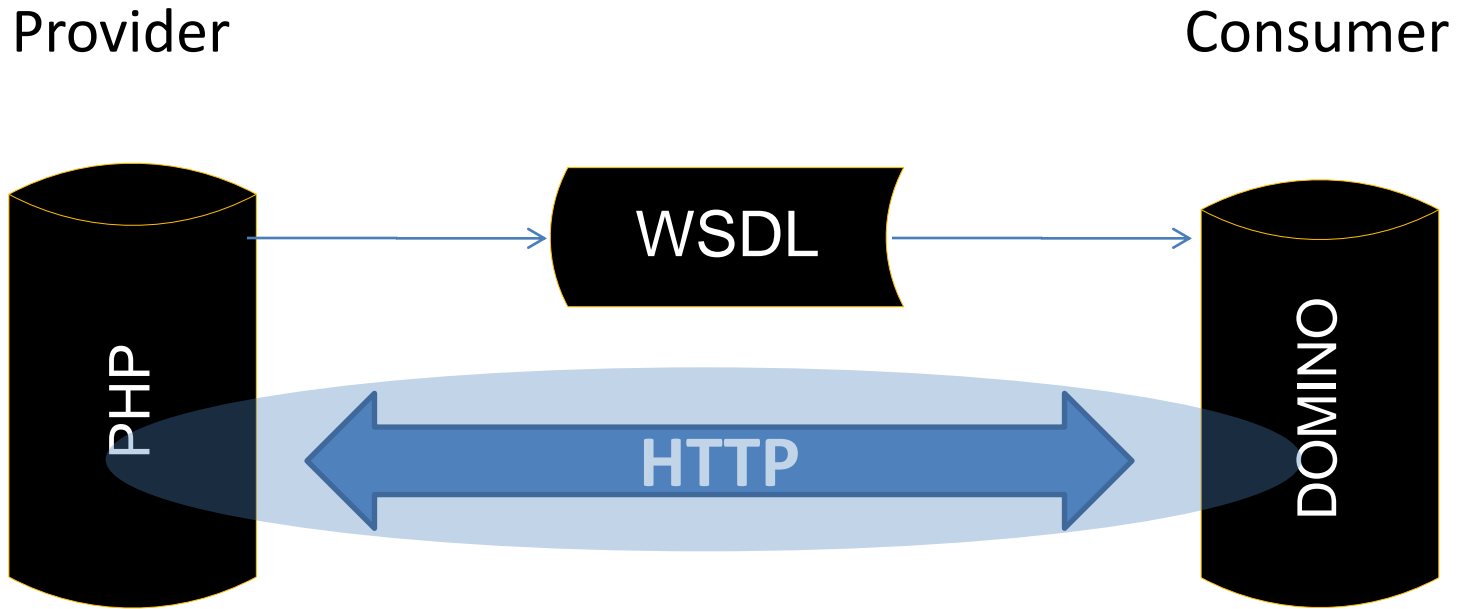


# Kommunikation – Datenaustausch per XML



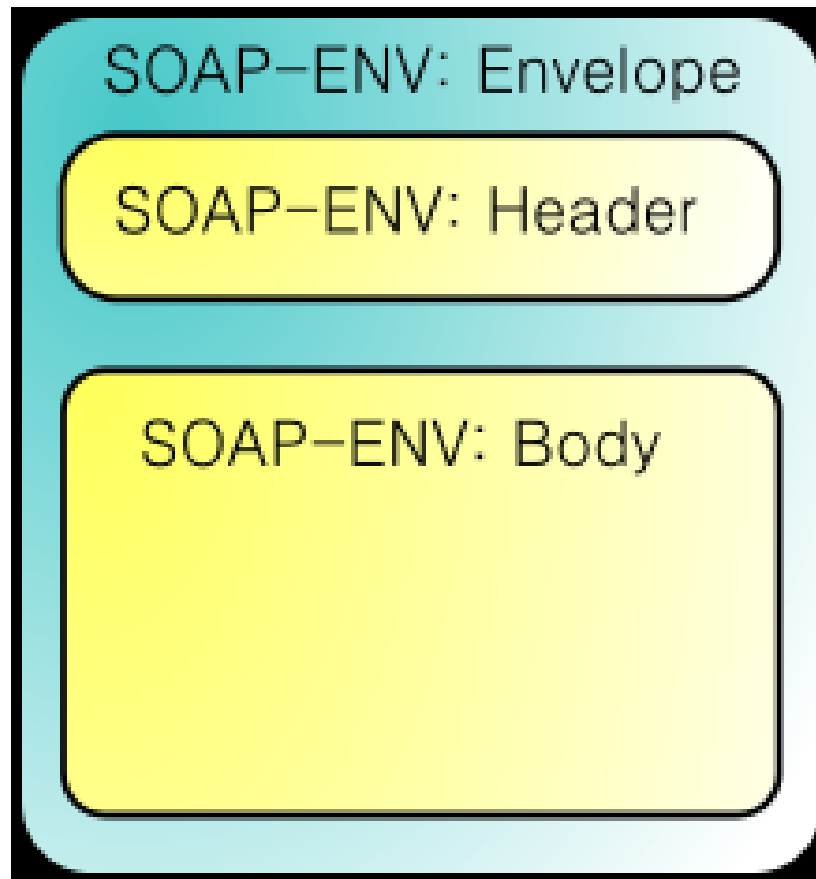
-Einfacher Funktionsaufruf mit Parameter aus PHP heraus

# Kommunikation – Datenaustausch per XML



-Einfacher Funktionsaufruf mit Parameter aus PHP heraus

# Kommunikation – Datenaustausch per XML



# Kommunikation – Datenaustausch per XML

```
<?xml version="1.0"?>  
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope">  
  <s:Body>  
    <m:TitleInDatabase xmlns:m="http://www.lecture-db.de/soap">  
      DOM, SAX und SOAP  
    </m:TitleInDatabase>  
  </s:Body>  
</s:Envelope>
```

# Kommunikation – Datenaustausch per XML

```
<?xml version="1.0"?>
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope">
  <s:Header>
    <m:RequestID xmlns:m="http://www.lecture-
db.de/soap">a3f5c109b</m:RequestID>
  </s:Header>
  <s:Body>
    <m:DbResponse xmlns:m="http://www.lecture-db.de/soap">
      <m:title value="DOM, SAX und SOAP">
        <m:Choice value="1">Arbeitsbericht Informatik</m:Choice>
        <m:Choice value="2">Seminar XML und Datenbanken</m:Choice>
      </m:title>
    </m:DbResponse>
  </s:Body>
</s:Envelope>
```

# DEMO

**Online-Shop  
Kaffeekochen**

# DEMO

**Authentication**

**Amazon**

# Give-Aways

- `ws.Setcredentials("User", "Password")`
  - Run as Web User
- `Ws.Setendpoint( „url“ )`
  - Lenkt den Webservice auf diese URL
  - Zur Laufzeit (nicht im Consumer Hardcodieren)
  - Danke Bob Balaban!



# Fragen

???

**Vielen Dank**

# Tim Pistor



Tim Pistor

Hornschuchstrasse 4

95336 Mainleus

Tel. : 09229 973260

timpistor@qmesh.de

Bleedyellow

Greenhouse

Skype (timpistor)

ICQ : 307-369-605

XING

Atnotes.de

## **Weiterführende Infos:**

- Designerhilfe (ab 8?)
- Lxsd.lss (im Notesverzeichnis)