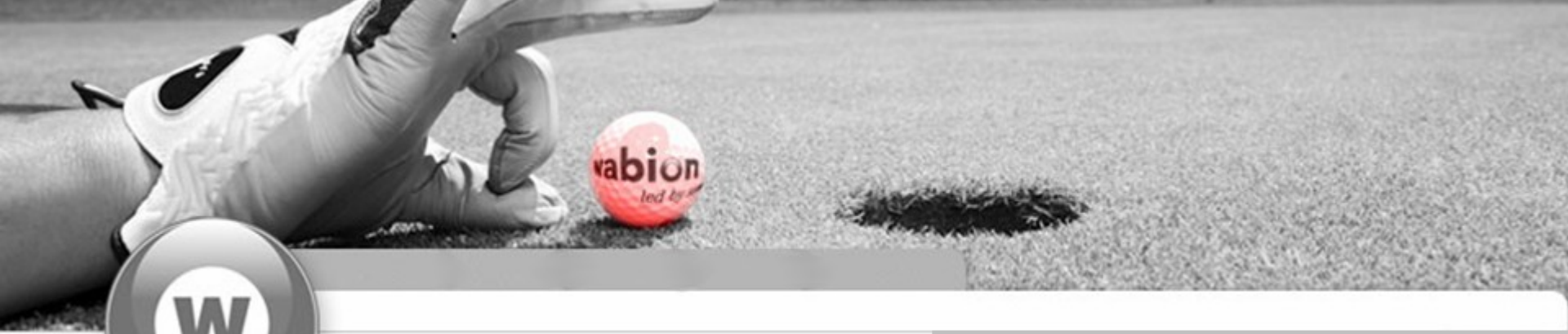


wabion

led by sense

Erstellung und Nutzung von REST Schnittstellen und Datenaustausch per JSON

Mathias Bierl
Wabion GmbH

ENTERPRISE PROFESSIONAL Google

NOTES & DOMINO ENTWICKLERCAMP

IBM Business Partner

Agenda

- Vorstellung
- Was sind REST Schnittstellen ?
- Aufbau von REST Schnittstellen
- Vor- und Nachteile von REST Schnittstellen
- Datentypen
- Einfache REST Schnittstellen
- Einbinden von REST Schnittstellen

Über Wabion

Gegründet 2004

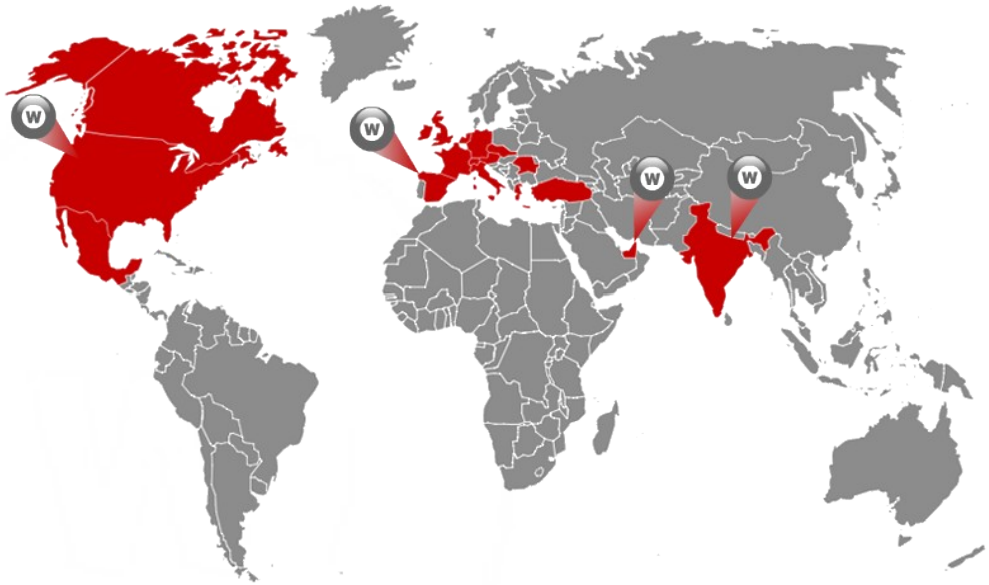
Firmensitz in Esslingen –

Niederlassungen in Österreich und Schweiz

Einsatz weltweit

15 feste Mitarbeiter

15 freie Mitarbeiter



Größter Google Enterprise Partner in Deutschland.

Focus Bereiche:

- Ganzheitliche Groupwarelösungen und deren Einsatz im Unternehmen (IBM und Google)
- Migrationen zwischen Groupwarelösungen und zu Cloud Lösungen
- Enterprise Search auf Basis Google Search Appliance
- Anwendungsentwicklung auf Basis Google Apps Engine und Lotus Domino
- Beratungsunternehmen mit dem Focus jeweils Technologieführer auf seinem Gebiet zu sein.

Wer bin ich ?

Mathias Bierl, Staatl. Gepr. Wirtschaftsinformatiker
(mathias.bierl@wabion.com)
Notesentwicklung / Administration seit 1997
(R 4 – ND 8.5.x)
Anti-Spam/Anti-Virus Lösungen
Enterprise Search/Knowledgemanagement
Google Lösungen (Search, Apps,…)
Server-Virtualisierung

Wer bin ich ?

- Projekte mit Lotus / Domino:
  - Lotus Notes R4 – ND 8.5.x
  - Sametime, Domino.Doc, Lotus Workflow, LEI, DECS, DGW
  - Formelsprache, Skript, Klassen, Java, JS, HTML, AJAX
  - Migration R4-R5-ND6-ND7-ND8, Microsoft Exchange,…
  - C-/C++-API, WIN 3.11-WIN 7, OS/2, AIX, AS/400, Linux, ...
  - Datenanbindungen an SAP, RDBS
  - XML/XSL, Servlets, Web Services
  - Mobile Datenanbindung (Blackberry, OBMG,…)
- Projekte mit Google:
  - Google Search Appliance
  - Google Message Security
  - Google Message Discovery
  - Google Apps inkl. Migrationen
  - Google Apps Engine
- Enterprise Search mit Lucidworks Fusion

Was sind REST Schnittstellen ?

- REST steht für REpresentational State Transfer
- REST ist ein Architekturansatz, der beschreibt wie verteilte Systeme miteinander kommunizieren können
- REST ist weder ein Protokoll noch ein Standard, RESTful Implementierungen nutzen aber standardisierte Verfahren wie HTTP(S), URI, JSON oder XML
- Generell ist selbst eine statische Web Seite REST-konform, sofern sich deren Format nicht ändert
- Wichtiges Prinzip für REST ist die Möglichkeit abgerufene Informationen automatisch verarbeiten zu können

Was sind REST Schnittstellen ?

- Entwicklung
  Das REST-Paradigma entwickelte sich aus dem 1994 von Roy Fielding entworfenen HTTP Object Model. Fielding entwickelte seine Idee von einem einheitlichen Konzept über die Jahre weiter, bis er 2000 den REST-Architekturstil im Rahmen seiner Dissertation veröffentlichte. Das Programmierparadigma der „RESTful Application“ wurde allerdings häufig falsch umgesetzt und findet eigentlich erst seit neuestem (Stand 2014) Anklang in der Welt des World Wide Web. In seiner Arbeit geht Fielding dabei auf die verschiedenen Anforderungen ein, die für die Webarchitektur wichtig sind.
  Quelle: https://de.wikipedia.org/wiki/Representational_State_Transfer

Was sind REST Schnittstellen ?

- Grundprinzipien
  - Client-Server Modell/Adressierbarkeit
    Server stellt einen Dienst bereit, der von beliebigen Clients genutzt werden kann und über eine eindeutige URI identifizierbar ist
  - Zustandslosigkeit
    Jede Anfrage an einen REST Service bzw. die Antwort an den REST-Client muss alle Daten enthalten, die benötigt werden um die Anfrage bzw. die Antwort zu verstehen und verarbeiten zu können.

Was sind REST Schnittstellen ?

- Grundprinzipien
  - Caching
    Clients können bereits angefragte Antworten speichern und erneut verwenden. Kennzeichnung der Informationen als cacheable/non-cacheable.
  - Einheitliche Schnittstelle
    REST-Services nutzen eine vom implementierten Dienst entkoppelte Schnittstelle, indem die Daten in ein standardisiertes Format gebracht werden

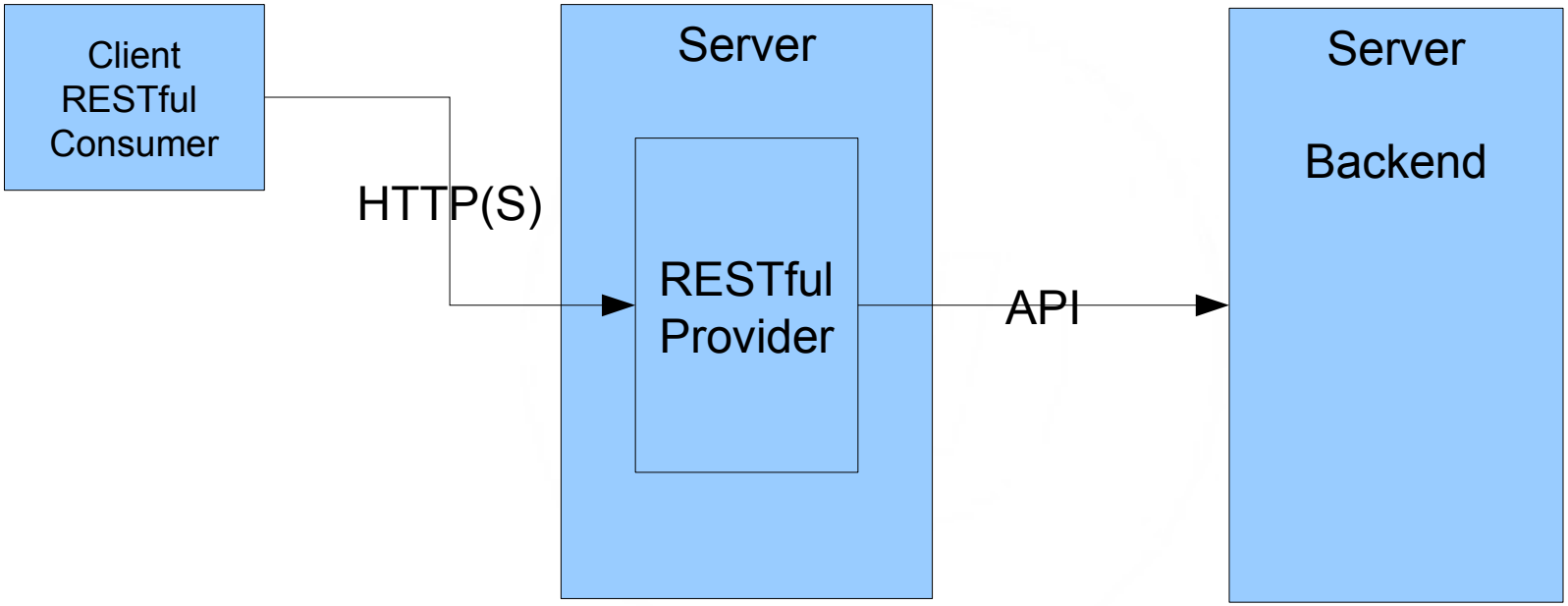
Was sind REST Schnittstellen ?

- Grundprinzipien
  - Mehrschichtige Systeme
    Systeme sind mehrschichtig und hierarchisch aufgebaut sowie streng voneinander abgegrenzt
  - Code on demand (optional)
    Code zur lokalen Ausführung wird erst im Bedarfsfall an den Client übermittelt.

Aufbau von REST Schnittstellen

- Die gängigsten REST Schnittstellen sind auf HTTP(S)-basierende Anfragen
- Diese können über die HTTP Methoden GET, POST, PUT, PATCH, HEAD und DELETE angeben, welche Operationen ausgeführt werden sollen
- Prinzipiell ist jede HTTP Anfrage bereits ein REST Service, auch wenn diese nur einzelne der Grundprinzipien erfüllt
-

Aufbau von REST Schnittstellen



Aufbau von REST Schnittstellen

- Jede REST Schnittstelle ist über eine eindeutige und langlebige URI erreichbar
- Dies bedeutet, daß sich die URI im Laufe der Zeit nicht ändert und immer die gleiche Datenstruktur zurückliefert
- Wenn sich eine Schnittstelle ändern kann, sollte diese von vorneherein versioniert werden, d.h. eine URI Komponente für die Version beinhalten
  z.B. http://meinserver/v1/user
- Dies führt dazu, daß es nicht der Fall sein darf, daß es eine URI gibt, die immer auf die aktuellste Version der API zeigen darf
- Es können auch Query-Parameter zur Ansteuerung der gewünschten Daten verwendet werden

Vor- und Nachteile von REST Schnittstellen

- Die Grundprinzipien von REST Schnittstellen führen zu bestimmten Vor- und Nachteilen
  - Vorteile
    - Portierbarkeit der Clients, da die Schnittstelle keine Abhängigkeit zum Client hat
    - Leichte Skalierbarkeit, da stateless Kommunikation
    - Vereinfachte Architektur, da Informationen in einem standardisierten vom eigentlichen Datenspeicher unabhängigen Format zurückgegeben werden

Vor- und Nachteile von REST Schnittstellen

- Nachteile
  - Nachteile bei der Netzwerkperformance, da immer alle Informationen mitgeliefert werden müssen
  - Server hat keine Kontrolle über ein konsistentes Verhalten der Client-Anwendung
  - Caching kann zur Nutzung von veralteten Informationen führen
  - Schlechtere Effizienz durch Umwandlung der Informationen in standardisierte Formate
  - Zusätzlicher Overhead und erhöhte Latenzen bei mehrschichtigen Systemen

Datentypen

- Theoretisch kann eine REST Schnittstelle Daten in jeglichem Format zurückliefern
- Gebräuchlich sind aktuell Plain Text, CSV, JSON, XML oder RSS
- Beispiele:
  - CSV: `Wert1;Wert2;Wert3;Wert4`
  - JSON: {“ID1“: “Wert1“,“ID2“: “Wert2“,“ID3“: “Wert3“,“ID4“: “Wert4“}
  - XML/RSS:

```
<?xml version="1.0" encoding="UTF-8"?>
<ID1>Wert1</ID1>
<ID2>Wert2</ID2>
<ID3>Wert3</ID3>
<ID4>Wert4</ID4>
```

Einfache REST Schnittstellen

- Wie können wir das jetzt einfach unter Domino nutzen ?
  - REST Schnittstellen zum Lesen von Daten können einfach über Notes Views abgebildet werden
    - $$ViewTemplate
      Definiert den Datentyp zur Rückmeldung und ggfls notwendige Rahmendaten, z.B. bei JSON oder XML
    - View
      HTML Ausgabe und Treat View as HTML contents Option

Einfache REST Schnittstellen - Beispiele

- Textbasierte REST Schnittstelle zur Abfrage von Personendaten aus dem Domino Directory
- CSV-basierte REST Schnittstelle zur Abfrage von Personendaten aus dem Domino Directory
- JSON basierte REST Schnittstelle zur Abfrage von Personendaten aus dem Domino Directory

Einfache REST Schnittstellen - Beispiele – Vergleich

- Text/CSV-basierte REST Schnittstelle
  - Leichte Auswertbarkeit, da nur minimale Interpretation nötig
  - Daten sind immer im gleichen Format und gleicher Reihenfolge
  - Mehrfachwerte sind schwerer zu realisieren

Einfache REST Schnittstellen - Beispiele – Vergleich

- JSON basierte REST Schnittstelle
  - Etwas komplexere Auswertbarkeit, dafür aber Strukturen möglich
  - Daten sind immer im gleichen Format, die Reihenfolge ist hierbei egal und es können leicht neue Daten hinzugefügt werden, ohne Beinträchtigung der Schnittstelle und bereits aktive Clients
  - Mehrfachwerte können einfach realisiert werden

Einbinden von REST Schnittstellen

- Eine REST Schnittstelle kann über wenige Zeilen Code angefragt werden.
- Nutzung von JavaScript Frameworks vereinfachen dies noch
-

Einbinden von REST Schnittstellen JavaScript

```
var myRequest = new XMLHttpRequest();
myRequest.open('GET', '/ec/names.nsf/getPerson?OpenView');
myRequest.onreadystatechange = function () {
    if (myRequest.readyState === 4) {
        alert(myRequest.responseText);
    }
};
myRequest.send();
```

- Funktioniert in allen Browsern außer IE < 7

Einbinden von REST Schnittstellen jQuery

```
$.ajax({
	url: '/ec/names.nsf/getPerson?OpenView',
	method: ‚GET‘
}).done(function (data, textStatus, jqXHR) {
	alert(data);
});
```

•

Einbinden von REST Schnittstellen Interpretation von JSON

- Bei JavaScript Frameworks kann man den erwarteten Rückgabetyp einstellen
  z.B. jQuery:
  ```
  dataType: "json"
  ```
- Bei klassischem JavaScript muss der Wert noch als JSON interpretiert werden
  ```
  var jsondata = JSON.parse(data);
  ```

Q & A

Kontakt

Wabion GmbH
Limburgstrasse 31
D-73734 Esslingen
Phone: +49 (0)711 – 25 25 52 24
Fax: +49 (0)711 – 25 25 52 14
Web: http://www.wabion.com

Mathias Bierl
eMail: Mathias.Bierl@wabion.com
Mobil: +49 (0) 172 877 27 65