Stefan Kühnlein

22. November 2012

REST Web Services – SOA Enablement für das Web

Agenda

REST Web Services – SOA Enablement für das Web

- Was ist REST?
- Ressource Orientierte Architekturen
- REST Schlüsselkonzepte
- RESTful Web Service
- Vergleich klassischer und moderner Architektur
- RESTful SOA

Was ist REST?

- Representational State Transfer
- REST wird in der Dissertation von Roy Fielding zum ersten mal beschrieben
  - http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
- REST ist ein Design Pattern und kein Protokoll
- Eine Neubetrachtung des Http-Protokolls unter dem Aspekt der Maninpulation von Ressourcen
- Beinhaltet architekturelle Elemente wie
  - Datenelemente (Ressourcen, Identifiers, Daten, Repräsentation, Kontrolldaten,…)
  - Connections (Client, Server, Cache)

Was ist REST?

Formaler ausgedrückt ist REST …

- … ein einfacher Weg zur Bereitstellung von Diensten für eine Client-Server Kommunikation basierend auf Ressourcen.
- … konsistent mit dem HTTP Standard und es ist relativ einfach Dienste bereit zu stellen und diese auch zu verwenden.
- … ein Standard für moderne Web 2.0 Anwendungen sowie für Rich-Client Anwendungen um mit „Back-End“ System zu interagieren
- In der Regel verwendet REST ein „leichtgewichtiges“ Datenmodel wie JSON

Ressource Orientierte Architektur

- Ressource Orientierte Architekturen basieren auf den Konzept von Ressourcen
- Ressourcen folgen einem relativ generischen Konzept
- Jede Ressource ist eindeutig identifizierbar und der Status kann über eine standardisierte Schnittstelle geändert werden.
- Eine Ressource zeichnet sich durch folgende Eigenschaften aus:
  - Ressource
  - Name der Ressource
  - Darstellung der Ressource
  - Link zur Ressource
  - Schnittstelle der Ressource

REST Schlüsselkonzepte
Ressource

http://www.doag.de/doag2012/ressource

„Ressource“: (Substantiv) eine Quelle von Informationen

Ressource

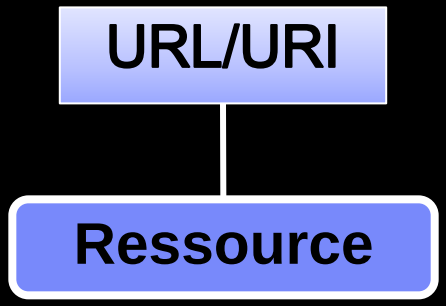
Eine Ressource kann jede beliebige Informationsquelle sein (z.B. Ein Konferenzvortrag, ein Teilnehmer der DOAG, …)

Eine Ressource kann aber auch z.B. eine informelle Beschreibung einer Person sein.

REST Schlüsselkonzepte
Ressourcen haben eine URI

http://en.wikipedia.org/wiki/Uniform_Resource_Locator

„A URL is technically a type of uniform resource identifier (URI) but in many technical documents and verbal discussions URL is often used as a synonym for URI"



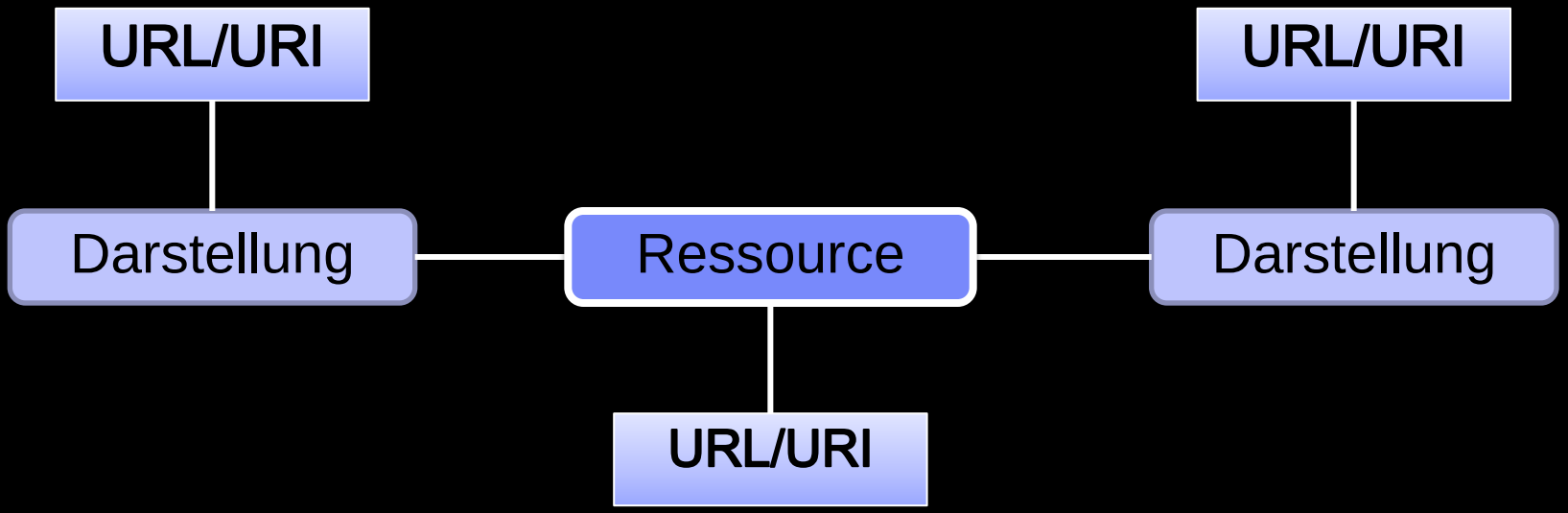
Uniform Resource Identifiers sind eindeutige Ids für Ressourcen wie die ISBN-Nummer eines Buches oder die Nummer eines Personalausweises

REST Schlüsselkonzepte
Ressourcen haben Darstellungen

Die Darstellung von Ressourcen kann in verschiedenen Formaten (XML, HTML, JSON, …) erfolgen.

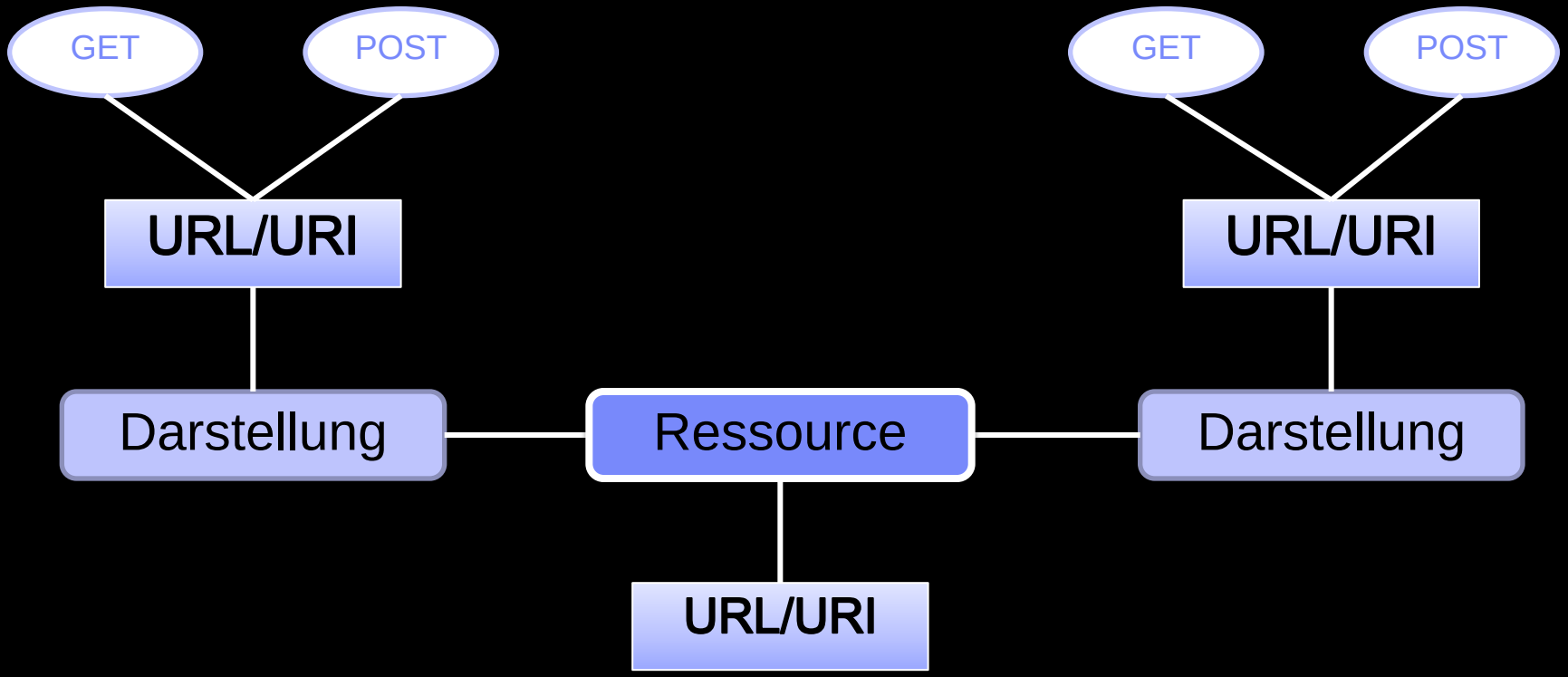
Die Darstellung haben auch Links zu anderen Ressourcen

REST Schlüsselkonzepte
Darstellungen haben ebenfalls URLs



URLs können eine spezielle Darstellung einer Ressource wiedergeben

REST Schlüsselkonzepte
Verwendung der Ressourcen



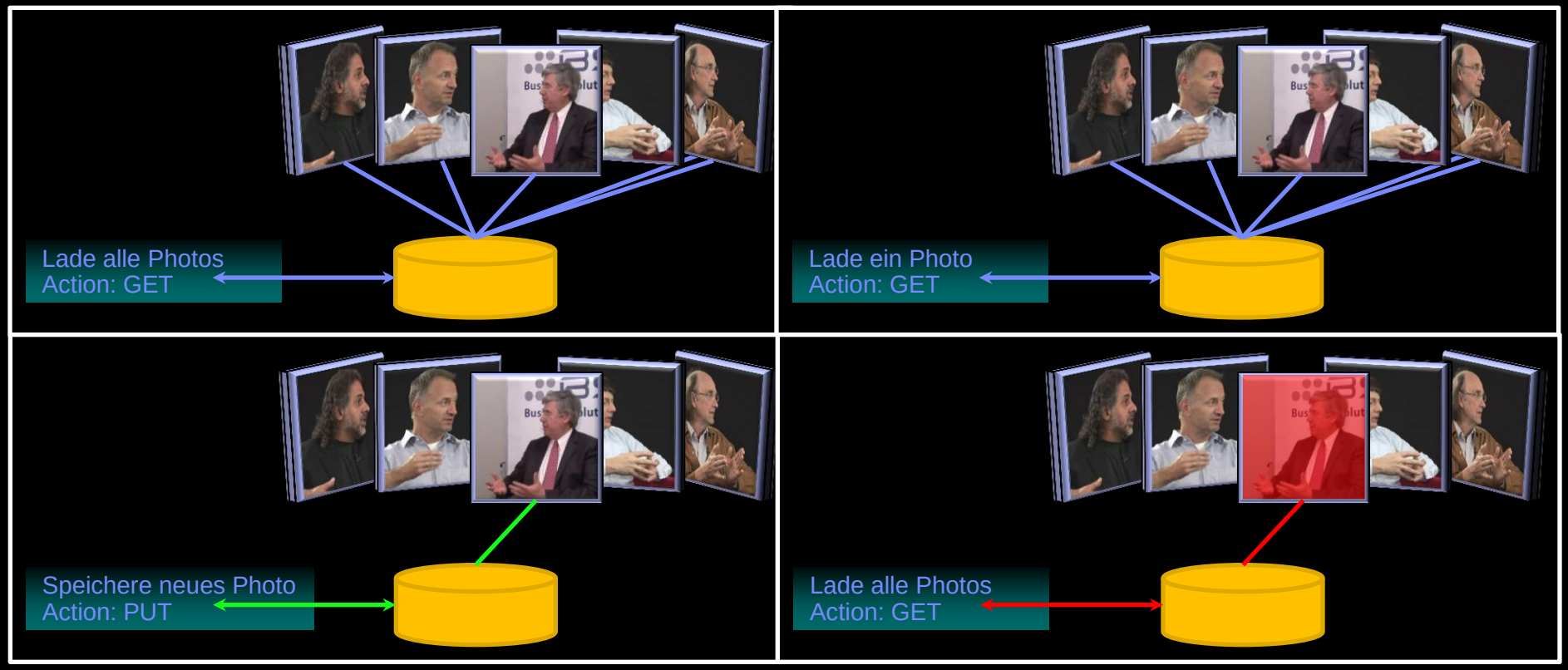
REST Schlüsselkonzepte
Verwendung der Ressourcen

- Die Schnittstelle für den Zugriff basiert Http-Protokoll
- Folgende Operationen stehen für die Bearbeitung von Ressourcen zur Verfügung

Ressource Methode	Beschreibung	Http-Operation
Create()	Erstellen einer neuen Ressource	PUT
GetRepräsentation()	Ermitteln einer Ressource	GET
Delete()	Löschen einer Ressource	DELETE; POST
Modify	Ändern einer Ressource	POST
GetMetaInformation()	Ermitteln der Metadaten einer Ressource	HEAD

RESTful Web Service

- Ein RESTful Web Service kann als „Satz“ formuliert werden
  - Verb = Http Aktion (Put, Get, Put, Delete)
  - Substantiv = URI des Service
  - Adjektiv = MIME-Type des resultierten Ergebnisses



RESTful Web Services
REST und dynamische Webprogrammierung

REST
Einfach verwendbar
Http-Pattern

RESTful Web Service

JSON / XML /…
Informationsaustausch
JavaScript „freundlich“

Anwendung A

Anwendung D

Anwendung D

Anwendung C

RESTful Web Services
Unterschied zwsichen REST und SOAP

- REST basiert auf Ressourcen und nicht auf Diensten
- Operationen basieren auf den Http-Methoden; keine Verwendung von URIs wie getCustomer()
- Verwendung von verschiedenen Datenformaten und nicht auf eine spezifische XSD fokussiert
- Der Fokus liegt auf der Bedeutung der URI
- Nutzt die Funktionen von Http wie z.B.
  - Content Negotiation
  - Security
- In der Regel wesentlich einfacher in der Verwendung und Erzeugung

RESTful Web Services
Missverständisse – Was REST nicht ist…

- REST ist die Verwendung von XML über http und nicht über SOAP
  - Nein! REST ist ein Pattern für den Austausch von Ressourcen
  - RPC ist nicht REST

- REST kann nur für CRUD (Create, Read, Update, Delete) Semantik verwendet werden
  - Nein! Eine Ressource kann alles sein; von einem Business Process Service bis zu einem Bild

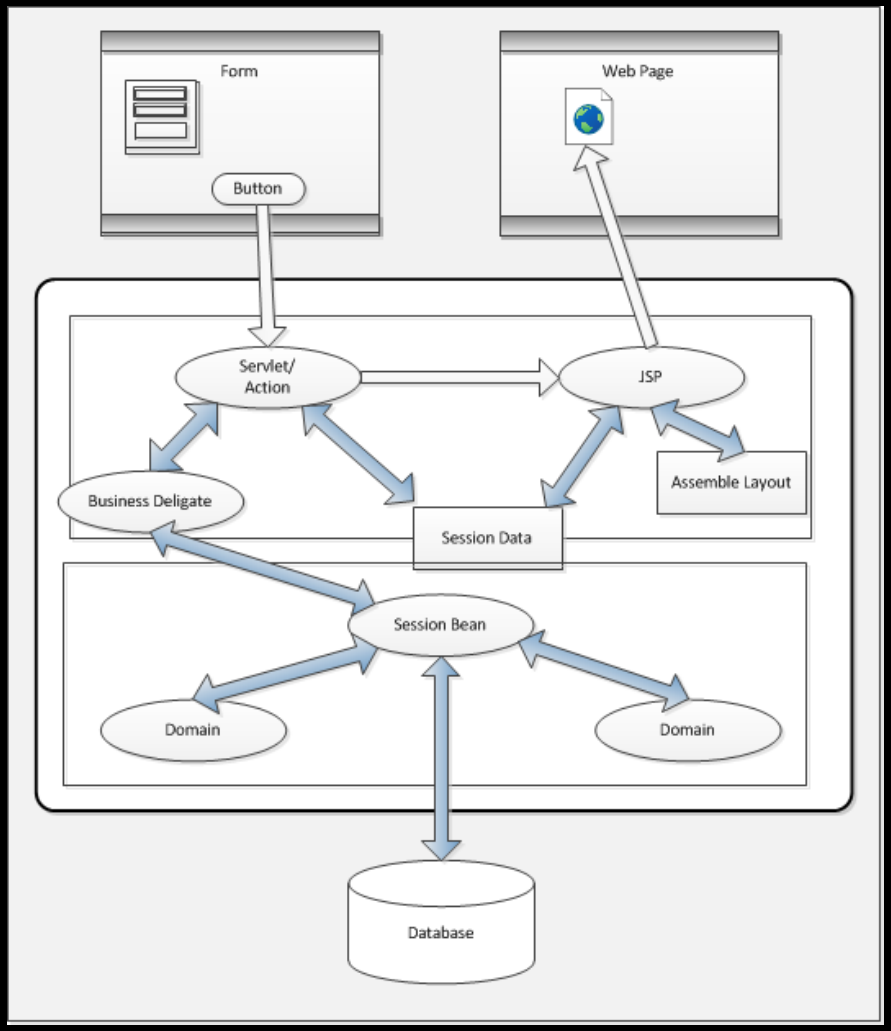
- Mit REST können traditionelle Web Services ersetzt werden
  - REST ist für den Zugriff auf Ressourcen über das Http-Protokoll
    - REST = Channel Exploitation
  - Mit REST bezieht sich nicht auf Transaktionen, Zuverlässigkeit und ist nicht geeignet für andere Protokolle oder Messaging
    - REST = Channel Abstaction

RESTful Web Services
Nachteile

- REST kann nicht auf jedes Problem angewendet werden
- Web 2.0 Clients verwenden ein einfach Modell bzgl. der Connecivity
  - Ein Web 2.0 Client kann nicht an einer globalen Transaktion teilnehmen
  - Ein Web 2.0 Client/Server ist begrenzt in der Quality of Service wie
    - Zuverlässigkeit
    - Vertraulichkeit
    - Integrität
  - Andere Enterprise Technologien bieten mehr Flexibilität bzgl.
    - Queues publish/subsribe
    - WS-Addressing und flexibles Routing
- Web 2.0 Interaktionen sind nicht die beste Lösung für
  - Bulk Transfer von Daten
  - Batch Verarbeitung
  - Lang laufende asynchrone Aufgaben

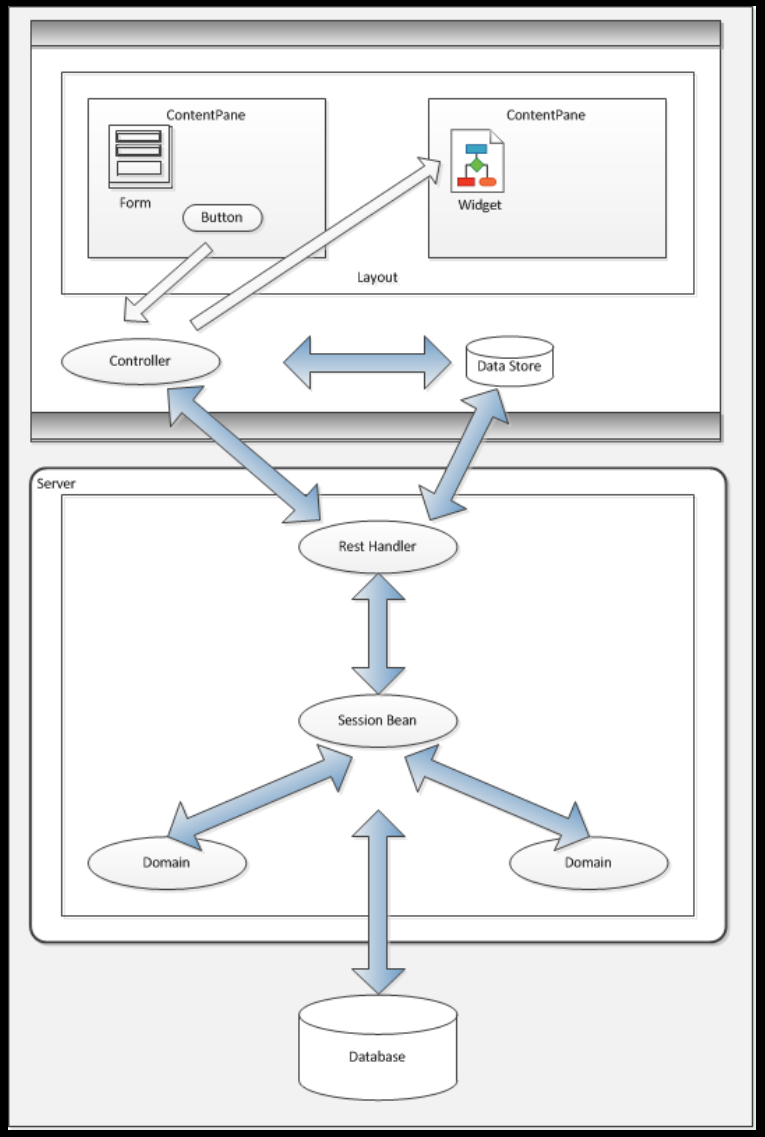
Klassische Web Architektur

- Server Rendering
  - Generate HTML
- Client State
  - Session Data
- Web Flows
  - Struts Forwards
- Layout Management
  - Assembling HTML Fragments
  - Tiles
  - Portlets



Moderne Web 2.0 Architektur

- Server Rendering
  - Data (JSON, XML)
- Layout Management
  - Layout Widgets
  - DOM Manipulation
    - Show/Hide
- Client State Management
  - Web State (im Browser)
    - Stores
    - Business State
  - Web Flow
    - Browser Script



RESTful SOA

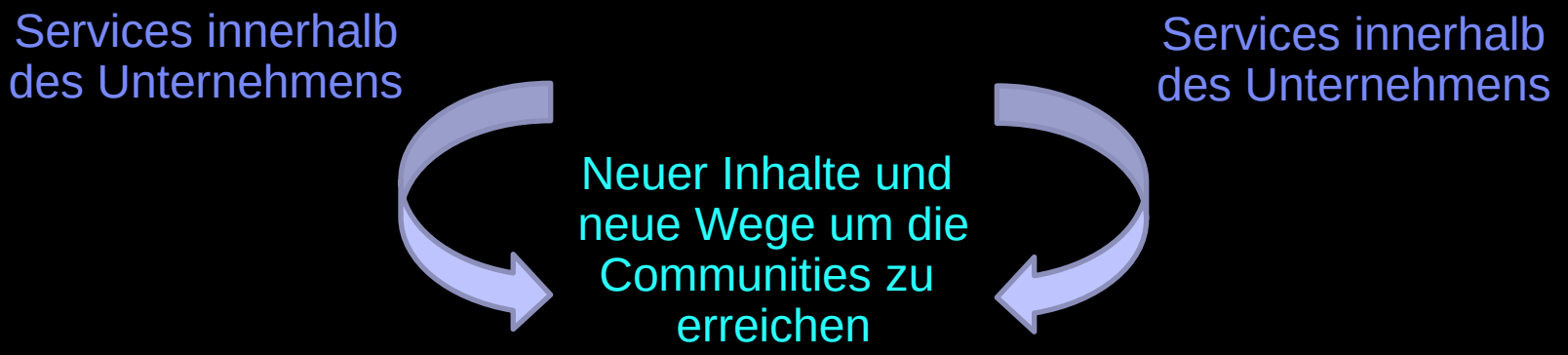
- RESTful SOA ist eine Art einer Service Orientierte Architektur, die die Konzepte des Webs als primäre Service Architektur verwenden
  - Primär wird REST für die Darstellung und den Zugriff auf die Ressourcen verwendet
  - Die Daten werden in JSON oder XML (incl. XML-Schema wie ATOM) übertragen
  - Unterstützt Rich User Interfaces die mit AJAX gebaut werden

- Schwerpunkte für den Aufbau einer effektiven RESTful SOA
  - Verwendung einer bestehenden Infrastruktur soweit wie möglich
  - Verwendung von bewährten Technologien für Skallierbarkeit, Performance und Sicherheit
  - Erstellung von Rich UIs die in herkömmlichen Browsern dargestellt werden können
  - Inhalte sollten leicht lesbar dargestellt werden

- Eine „herkömmliche“ Service Orientierten Architektur kann jeden Dienst anrufen und kann daher leichter implementiert werden

RESTful SOA
Alternative Verwendung von REST und SOA

- Eine typische Enterprise SOA nutzt traditionelle Standards wie SOAP
- Wenn die Entscheidung sowohl für RESTful SOA und Enterprise SOA gefallen ist, dann kann auf die Vorteile beider Konzepte zugegriffen werden
  - Services, die innerhalb ses Unternehmens genutzt werden (i.d.R. SOAP basiert)
  - Services, die außerhalb des Unternehmens genutzt werden (i.d.R. REST basiert)
- Dieses Konzept vereinigt das „Beste“ aus beiden Welten
- Technologien wie SCA kann hierbei helfen, die beiden Programmiermodelle zusammen zu bringen.



Danke für Ihre Aufmerksamkeit

Stefan Kühnlein
Advisory IT Architekt

IBM Deutschland
Hollerithstr. 1
81829 München
Mobil: +(49)160-8848611
Stefan.Kuehnlein@de.ibm.com