REST - eine erste Vertiefung

REST: eine erste Vertiefung- Ziele

- Die verschiedenen Arten von Ressourcen erklären können und an einem konkreten Beispiel verschiedene Möglichkeiten des Ressourcen Design abwägen können.
- Wesentliche Aspekte des URI Design erklären können und die Adäquatheit von konkreten URI Designs diskutieren können.
- Die Konzepte *Content Negotiation* und *Media Type* erklären können
- die Relevanz der Verwendung von *HTTP Statuscodes* erklären können
- das *Reife Modell für Rest Services* erklären können und die verschiedenen Stufen mit Beispielen hinterlegen können.

Ressourcen und Repräsentationen

- Eine Entität, die identifizierbar ist und eine oder mehrere Repräsentationen hat.
- Ressourcen mit mehreren Repräsentationen können auch als mehrere Ressourcen betrachtet werden (fließende Grenze)
- Die sinnvolle Definition von Ressourcen steht im Mittelpunkt der Definition einer Architektur im Rest Stil (ROA)

Primärressourcen

- Die bei der Betrachtung einer Domäne offensichtlich zu modellierenden Entitäten

  `http://example.com/customers/1234`
  `http://example.com/reminders/275983`

- Die Art der Implementierung einer Ressource darf für den Browser oder die Anwendung nicht erkennbar sein.

- Ressourcen sollen daher aus Sicht der Anwendung, nicht aus der der Implementierung definiert werden.

Subressourcen

- .. sind Teile anderer Ressourcen
- Entscheidung ob ein Element einer Entität als eigene (Sub-)Ressource modelliert wird oder in die Repräsentation der Ressource integriert wird

Listenressourcen

- In der Regel sollen neben den Primärressourcen auch **Listen von Primärressourcen** als Ressourcen modelliert werden.
- Als Reaktion auf GET wird eine Repräsentation der Liste übermittelt; über POST kann eine **neue Primärressource** als Element der Liste angelegt werden.

Filter und Projektionen

- Aus Gründen der **Effizienz** ist es oft sinnvoll nicht die komplette Repräsentation einer Ressource zu übertragen.
- Über Filter können Elemente einer Liste selektiert werden
- Über Projektionen können Informationen aus der Repräsentation einer Primärressource ausgewählt werden.

Paginierung

- Da vor allem Listenressourcen sehr große Repräsentationen haben können, ist es sinnvoll mit einem einzelnen GET nur einen Teil (eine „Seite“) der Repräsentation zu erhalten.

Konzeptressourcen

- Manchmal ist es sinnvoll neben verschiedenen Informationsressourcen auch ein gemeinsames, abstraktes Konzept benennen zu können.
- Beispiel: Wenn eine Visitenkarte, ein Bild, ein Lebenslauf einer Person Ressourcen sind, kann es sinnvoll sein die Person selbst als Konzeptressource zu modellieren.
- Auf ein GET auf eine Konzeptressource hin wird keine Repräsentation übermittelt sondern der Statuscode 303 „See Other“

URI Design

- URI bezeichnen Ressourcen, also „Dinge“, nicht Tätigkeiten (wie z.B. bei Methodennamen).

- `http://example.com/customers/{id}` erscheint sinnvoll

- `http://example.com/customers/create?name=XYZ` deutet auf ein Architekturproblem hin.

- `http://example.com/RequestProcessor?method=build&p1=x&p2=y` :

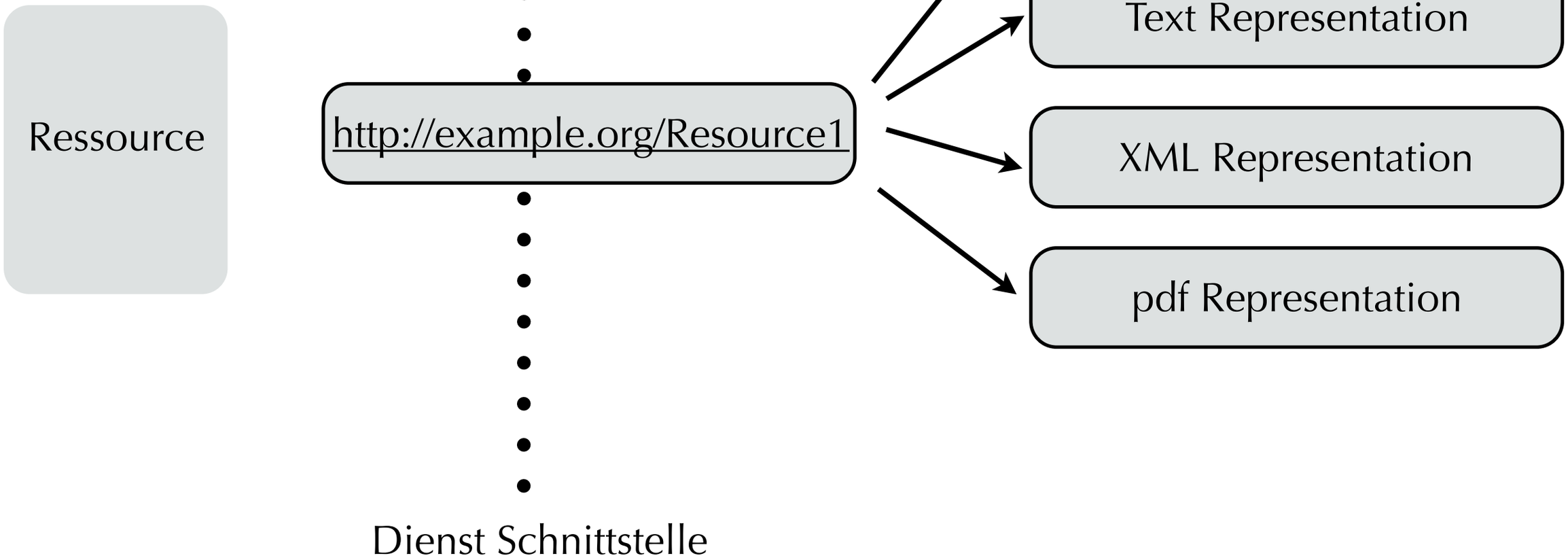
  hier scheint REST als Gateway zu einem objektorientierten System verwendet zu werden

- URI sollte Substantive, keine Verben enthalten!

Stabile URIs

- „Cool URIs don't change“ (Tim Berners-Lee)
- Eine Ressource kann mehrere URIs haben.
- Bei Änderungen kann über ein `Redirect` umgeleitet werden.
- Nicht mehr unterstützte URIs können auf http-Anfragen mit Statuscode `410 – Gone` reagieren (Anstelle von `404 Not Found`)

URI ist unabhängig von der Repräsentation



Content Negotiation

Ressource

http://example.org/Resource1

```
GET /Resource1 HTTP/1.1
Accept: application/xml
...
```

```
HTTP/1.1 200 OK
.....
Representation of Resource1
```

Dienst Schnittstelle

Dienst Nutzer

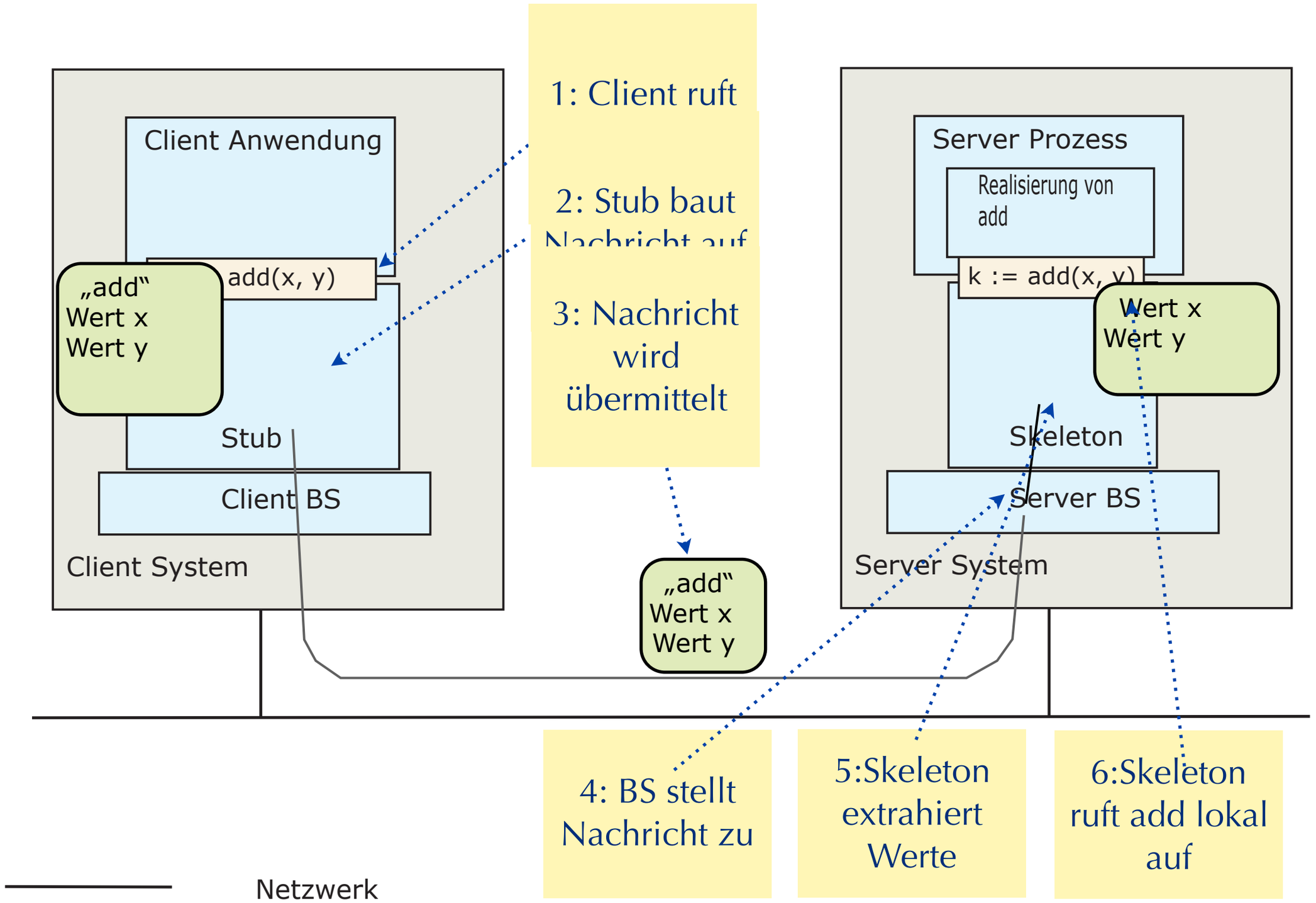
IANA Media Types

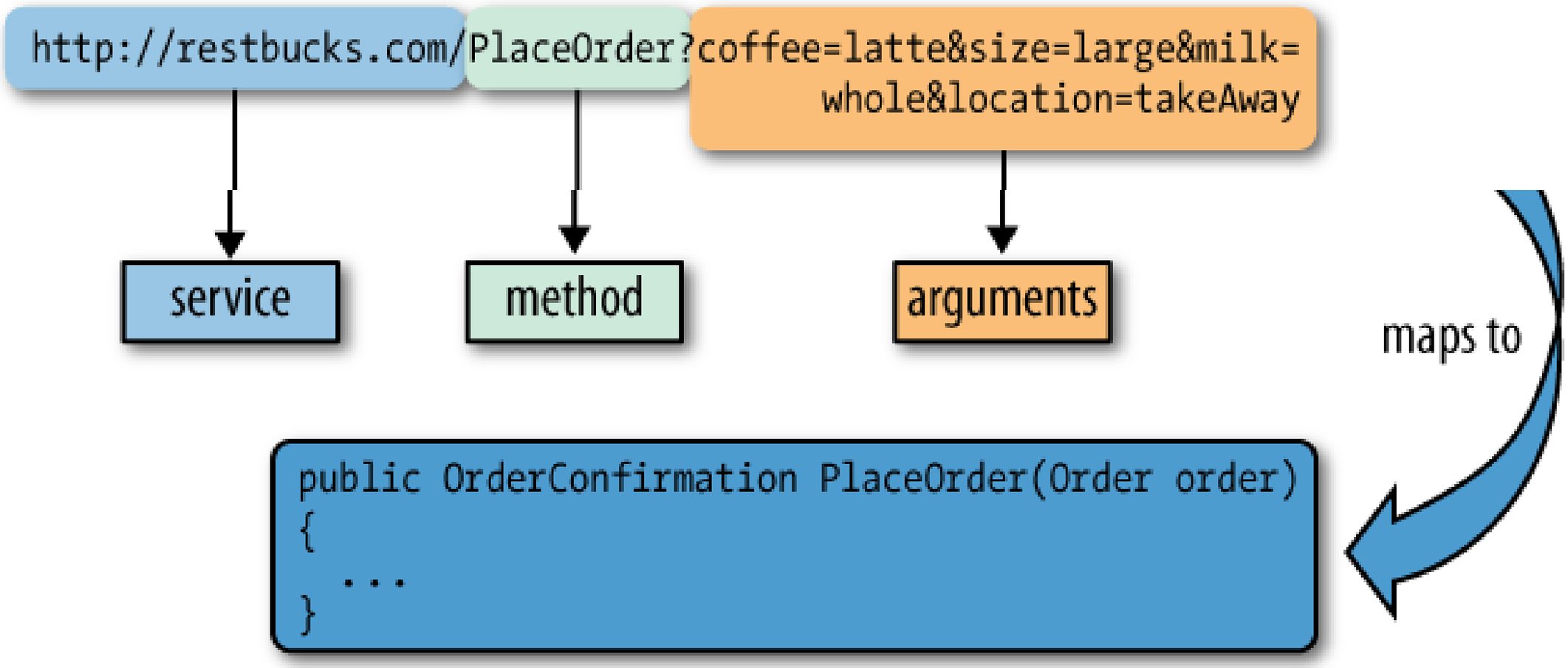
http://www.iana.org/assignments/media-types/media-types.xhtml

HTTP Status Codes

https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

Schritte bei der Ausführung eines RPC





URI Tunneling

client application domain

```
POST /PlaceOrder HTTP/1.1
Content-Type: application/xml
Host: restbucks.com
Content-Length: 361

<Order xmlns="http://
restbucks.com">
  <Location>takeAway</Location>
  <Item>
    <Name>latte</Name>
    <Quantity>1</Quantity>
    <Milk>whole</Milk>
    <Size>small</Size>
  </Item>
  <Item>
    <Name>cookie</Name>
    <Kind>chocolate-chip</Kind>
    <Quantity>2</Quantity>
  </Item>
</Order>
```

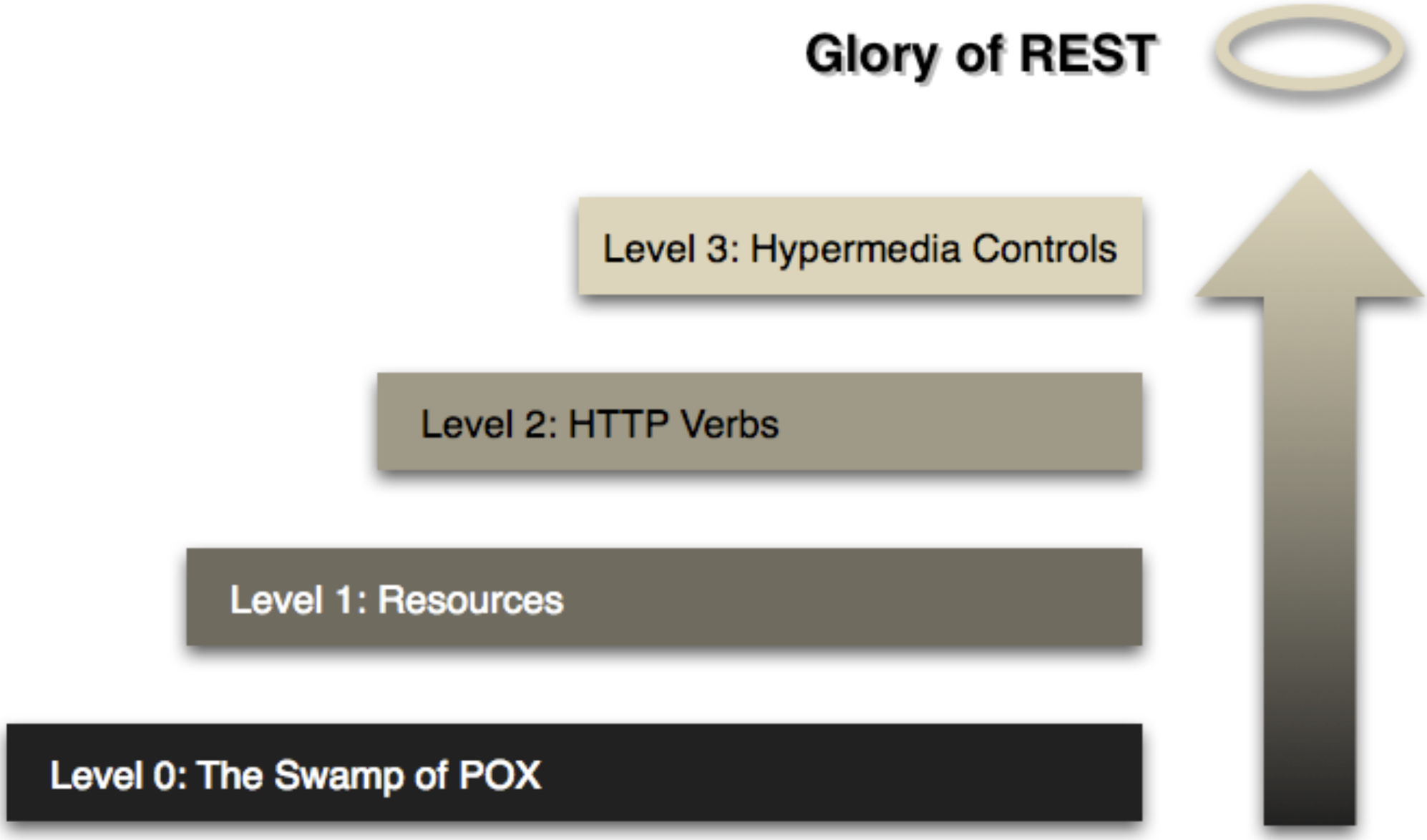
restbucks application domain

```
HTTP/1.1 200 OK
Content-Length: 93
Content-Type: application/xml;
charset=utf-8
Server: Microsoft-HTTPAPI/2.0
Date: Mon, 04 Aug 2008
18:16:49 GMT

<OrderConfirmation xmlns="http://
restbucks.com">
  <OrderId>1234</OrderId>
</OrderConfirmation>
```

POX

Richardson Maturity Model



Quelle:http://martinfowler.com/articles/richardsonMaturityModel.html, zuletzt abgerufen: 25.4.2016

REST: eine erste Vertiefung - Zusammenfassung

- Arten von Ressourcen
- Wesentliche Aspekte des URI Design
- Die Konzepte *Content Negotiation* und *Media Type*
- *HTTP Statuscodes*
- das *Reife Modell für Rest Services* (Richardson Maturity Model)