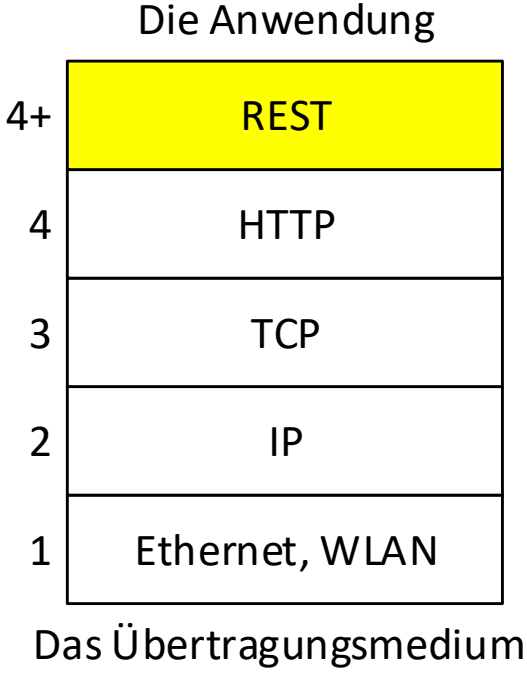
REST
Representational State Transfer

Was ist REST?

- Representational State Transfer (REST) bezeichnet ein Programmier-paradigma für verteilte Systeme, insbesondere für Webservices.
- Ein Webservice ermöglicht die Maschine-zu-Maschine-Kommunikation auf Basis von HTTP oder HTTPS über Rechnernetze.
- Dabei werden Daten ausgetauscht und auf entfernten Computern Funktionen aufgerufen.
- Jeder Webservice besitzt einen URI, über den er eindeutig identifizierbar ist, sowie eine Schnittstellen-beschreibung, die definiert, wie mit dem Webservice zu interagieren ist.
- Die Kommunikation erfolgt typischerweise über Protokolle aus dem Internetkontext wie
  - XML oder
  - JSON.



Was ist REST?

- REST hat das Ziel, einen Architekturstil zu schaffen, der die Anforderungen des modernen Web besser darstellt.
- Dabei unterscheidet sich REST vor allem in der Forderung nach einer einheitlichen Schnittstelle von anderen Architekturstilen.
- REST stellt eine einfache Alternative zu ähnlichen Verfahren wie SOAP und WSDL und dem verwandten Verfahren RPC dar.
- Anders als bei vielen verwandten Architekturen kodiert REST keine Methodeninformation in den URI, da der URI Ort und Namen der Ressource angibt, nicht aber die Funktionalität, die der Web-Dienst zu der Ressource anbietet.

Was ist REST?

- Der Vorteil von REST liegt darin, dass im WWW bereits ein Großteil der für REST nötigen Infrastruktur, wie
  - Web- und Application-Server,
  - HTTP-fähige Clients,
  - HTML- und XML-Parser sowie
  - Sicherheitsmechanismen

  vorhanden ist.
- Eine Ressource kann dabei über verschiedene Medientypen dargestellt werden, auch Repräsentation der Ressource genannt.
- Die Bezeichnung „Representational State Transfer“ soll den Übergang vom aktuellen Zustand zum nächsten Zustand einer Applikation verbildlichen.
- Dieser Zustandsübergang erfolgt durch den Transfer der Daten, die den nächsten Zustand repräsentieren.

REST ist zustandslos wie HTTP

- Jede REST-Nachricht enthält alle Informationen, die für den Server bzw. Client notwendig sind, um die Nachricht zu verstehen.
- Weder der Server noch die Anwendung soll Zustandsinformationen zwischen zwei Nachrichten speichern.
- Man spricht daher von einem zustandslosen Protokoll.
- Jede Anfrage eines Clients an den Server ist insofern in sich geschlossen, als sie sämtliche Informationen über den Anwendungszustand beinhaltet, die vom Server für die Verarbeitung der Anfrage benötigt werden.

Frontend 1: REST HTTP GET über einen Webbrowser...

https://frank-dopatka.de/tradingtool/getKursreihe/DE0007100000/01.01.2018

JSON Rohdaten Kopfzeilen

Speichern Kopieren Alle einklappen Alle ausklappen (langsam) JSON durchsuchen


```
kurse:
  0:
    o:      70.74
    c:      70.62
    h:      71.08
    l:      69.33
    datum:  "02.01.2018"
    vol:    279901184
  1:
    o:      70.75
    c:      71.19
    h:      71.55
    l:      70.68
    datum:  "03.01.2018"
    vol:    207582320
```


- Im Internet testbar unter https://frank-dopatka.de/tradingtool/

Frontend 1: REST HTTP GET über einen Webbrowser...

https://frank-dopatka.de/tradingtool/getKursreihe/DE0007100000/01.01.2018

JSON Rohdaten Kopfzeilen

Speichern Kopieren Alle einklappen Alle ausklappen (langsam) JSON durchsuchen


```
kurse:
  0:
    o:      70.74
    c:      70.62
    h:      71.08
    l:      69.33
    datum:  "02.01.2018"
    vol:    279901184
  1:
    o:      70.75
    c:      71.19
    h:      71.55
    l:      70.68
    datum:  "03.01.2018"
    vol:    207582320
```


Der Dienst hat 2 Input-Parameter:
...von der Daimler AG ab dem 01.01.2018

Aufruf des Dienstes:
Gib mir die Börsenkursreihe...

Reload = Polling
Gibt es etwas Neues?

- Im Internet testbar unter https://frank-dopatka.de/tradingtool/

Ein REST Backend mit PHP und Apache

- https://frank-dopatka.de/tradingtool/getKursreihe/DE0007100000/01.01.2018
- In der .htaccess im tradingtool-Ordner wird jede Anfrage auf die https://frank-dopatka.de/tradingtool/index.php geleitet...

```
<IfModule mod_rewrite.c>
  RewriteEngine On
  RewriteRule (.*) index.php
</IfModule>
```

Ein REST Backend mit PHP und Apache

- In der index.php wird zunächst die URI zerlegt...

```
$uri = $_SERVER['REQUEST_URI‘];   // http://localhost/tradingtool/
                                  // getKursreihe/
                                  // DE0007100000/01.01.2018

$uri_teile = explode('/',$uri);   // trenne am Backslash

$rest_dienst = $uri_teile[2];                          // getKursreihe

$rest_parameter = array_slice($uri_teile,3);           // Array mit...
                                                       // DE0007100000
                                                       // 01.01.2018

$rest_parameter_anzahl = count($rest_parameter);       // 2
```

Ein REST Backend mit PHP und Apache

- Nun wird der konkrete Dienst und die Anzahl der Parameter gesucht...

```
switch($rest_dienst){
...
case 'getKursreihe':
  if ($rest_parameter_anzahl==1){
  ...
  }
  else if ($rest_parameter_anzahl==2){ // DE0007100000 und 01.01.2018
    $db = (new DB())->getConnection(); // Verbindung zur MySQL-DB
    $isin = $rest_parameter[0];
    $von = $rest_parameter[1];
    header('Content-Type: application/json'); // Typ der Antwort
    echo(Server::getKursreihe($db,$isin,$von,null));
  }
  else ...{
  ...
  }
...
}
```

Ein REST Backend mit PHP und Apache

- Die Kursreihe holt die Daten der AG aus der MySQL-Datenbank sowie die benötigten Kurse...

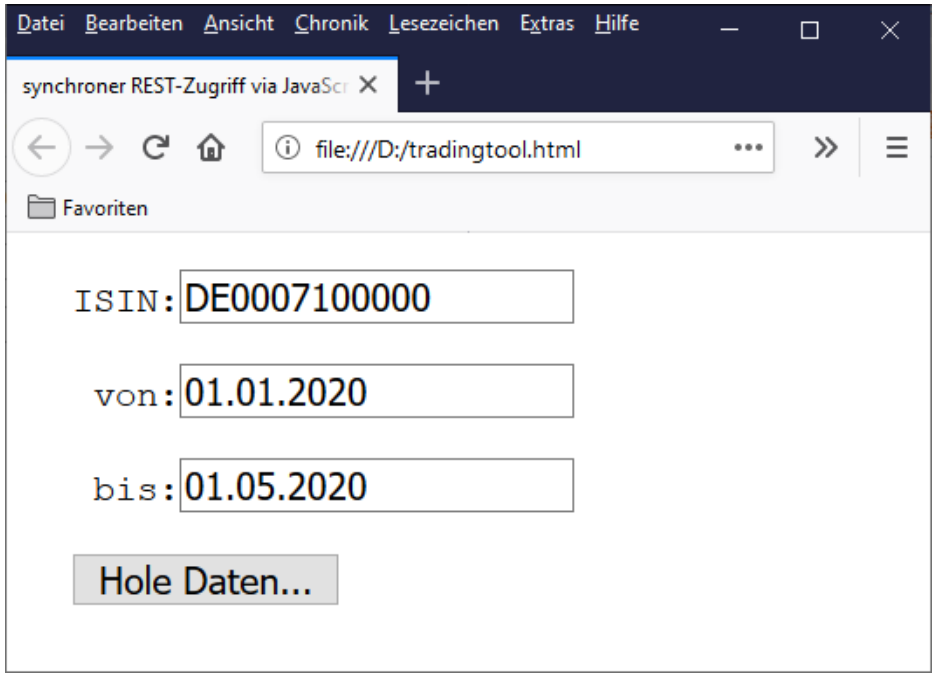
```
public static function getKursreihe($db,$isin,$von,$bis){
  $stmt = $db->prepare('SELECT * FROM kursreihe WHERE isin=?;');
  $stmt->bindParam(1,$isin); // Prepared Statement
  $stmt->execute();
  $row = $stmt->fetch(PDO::FETCH_ASSOC);
  $kr = new Kursreihe($db,$row); // Kursreihe erzeugen...
  Server::getKurse($db,$kr,$von,$bis); // ...und mit Kursen füllen
  return $kr->toString();
}
```

- ...und gibt dies im JSON-Format zurück:

```
public function toString(){
  return json_encode($this,JSON_PRETTY_PRINT|JSON_UNESCAPED_SLASHES);
}
```

Frontend 2:
Eine Webseite mit JavaScript...

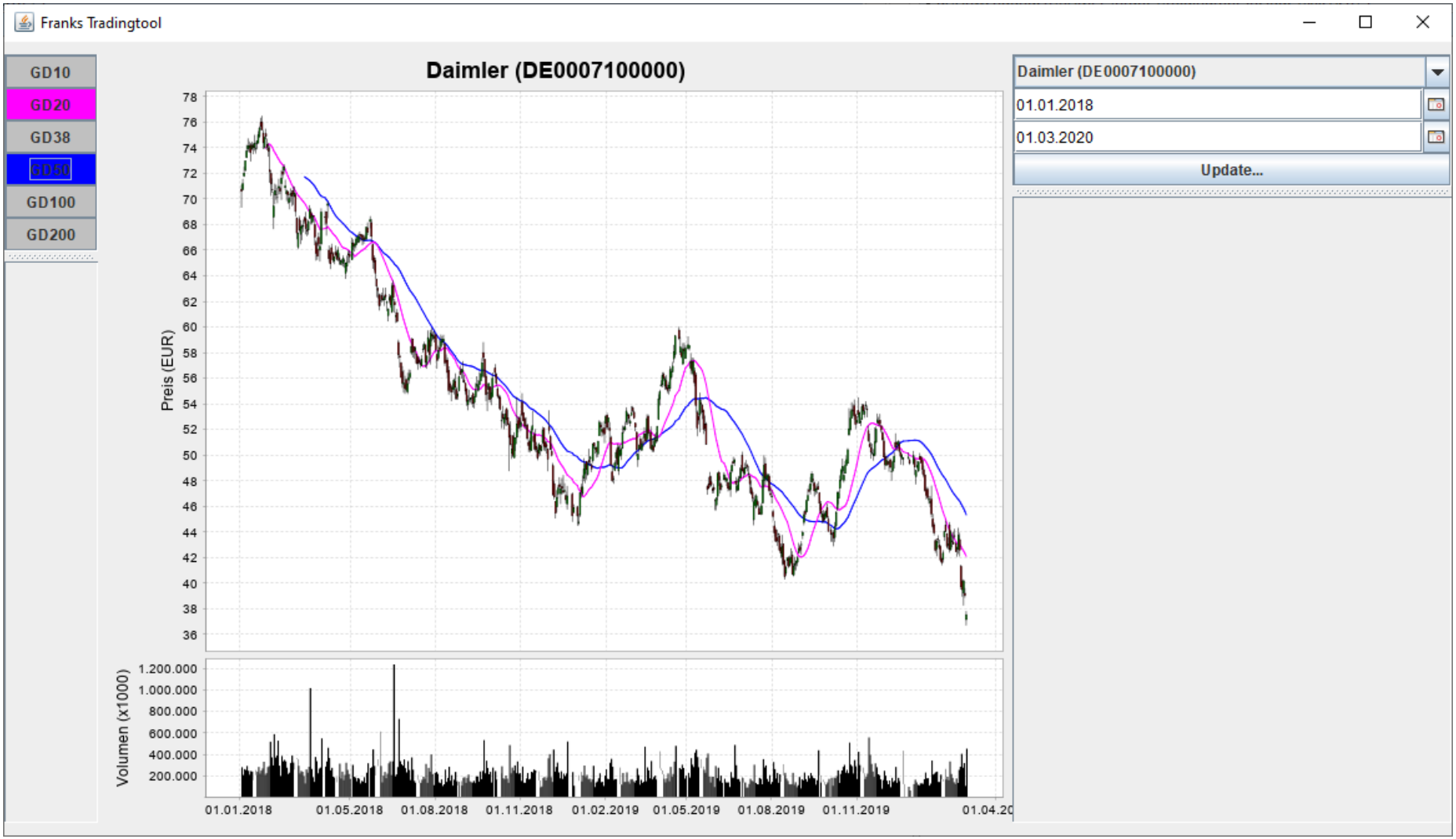
```
<body>
<pre>
  <label for="ISIN">ISIN:</label><input id="ISIN"/><br/>
  <label for="von">  von:</label><input id="von"/><br/>
  <label for="bis">  bis:</label><input id="bis"/><br/>
  <button type="submit" onclick="holeDaten()">Hole Daten...</button>
</pre>
<div id="zeigeDaten"></div>
</body>
```



Frontend 2:
Eine Webseite mit JavaScript...

```
<script>
function holeDaten() {
  var isin = document.querySelector("#ISIN").value;
  var von = document.querySelector("#von").value;
  var bis = document.querySelector("#bis").value;
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      var json = this.responseText;
//      alert(json);
      var kr=JSON.parse(json);
      var s="<p>ISIN: "+kr.isin+"</p>";
      s+="<p>Name: "+kr.name+"</p>";
      s+="<p>Kurse ab dem "+kr.von+"</p>";
      s+="<p>Anzahl der Kurse: "+kr.kurse_anzahl+"</p>";
      for(var i=0;i<kr.kurse.length;i++){
        s+="<p>Schlusskurs vom "+kr.kurse[i].datum+" war "+kr.kurse[i].c.toFixed(2)+" "+kr.einheit+"</p>";
      }
      var container = document.getElementById("zeigeDaten");
      container.innerHTML = s;
    }
  };
  // false: synchrone Anfrage
  xhttp.open("GET","https://frank-dopatka.de/tradingtool/getKursreihe/"+isin+"/"+von+"/"+bis,false);
  xhttp.send();
}
</script>
```

Frontend 3:
Java Swing & JFreeChart...



Frontend 3: Java Swing & JFreeChart...

- Aufruf des REST-Dienstes und Konvertierung der JSON-Daten in ein Java-Objekt...

```
import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
public Kursreihe getKursreihe(String isin) {
  String json = client.target(url+"getKursreihe/"+isin).request().
accept("application/json").get(String.class);
  return (Kursreihe) fromJSON(json,Kursreihe.class);
}
```

- ...mit der Hilfsmethode der Konvertierung unter Verwendung von GSon:

```
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
@SuppressWarnings("unchecked")
private static Object fromJSON(String json,
@SuppressWarnings("rawtypes") Class c) {
  Gson gson = new GsonBuilder().setPrettyPrinting().create();
  return gson.fromJson(json,c);
}
```

Frontend 4: Microsoft Excel...

	A	B	C
1			
2		ISIN:	DE0007100000
3		von:	01.01.2018
4		bis:	28.02.2020
5			
6		Download...	

	A	B	C	D	E	F
1	Name:	Daimler				
2	ISIN:	DE0007100000				
3	von:	02.01.2018				
4	bis:	28.02.2020				
5	Einheit:	EUR				
6	Handelsplatz:	Xetra				
7	ist Index:	nein				
8	ist ETP:	nein				
9	ist Devise:	nein				
10	datum	open	close	high	low	volumen
11	02.01.2018	70,74	70,62	71,08	69,33	279901184
12	03.01.2018	70,75	71,19	71,55	70,68	207582320
13	04.01.2018	71,8	72,02	72,5	71,74	262169152
14	05.01.2018	72,25	72,98	72,98	72,16	254139248
15	08.01.2018	73,24	73,75	74,21	73,21	291938912
16	09.01.2018	73,9	74,16	74,71	73,68	246767696
17	10.01.2018	74	74,15	74,48	73,72	210212128
18	11.01.2018	74,2	73,61	74,36	72,94	244357792
19	12.01.2018	73,93	74,18	74,46	73,8	238663440
20	15.01.2018	74,2	74,01	74,31	73,51	159911808

Frontend 4: Microsoft Excel...

```
Sub Schaltfläche1_Klicken()
Dim request As Object
Dim response As String
Dim link As String
Dim zeilen, zellen As Variant
Dim zeilenAnzahl, zellenAnzahl As Long
Dim i, j As Long
Set request = CreateObject("MSXML2.XMLHTTP")
link = "https://frank-dopatka.de/tradingtool/getKursreiheCSV/"
link = link + Range("C2").Text + "/" + Range("C3").Text + "/" + Range("C4").Text
With request
    .Open "GET", link, False 'HTTP REST GET Request
    .SetRequestHeader "Content-Type", "text/csv"
    .Send 'absetzen...
    While request.readyState <> 4
        DoEvents 'warten, bis das ganze Ergebnis da ist
    Wend
    response = .ResponseText
End With
```

	A	B	C
1			
2		ISIN:	DE0007100000
3		von:	01.01.2018
4		bis:	28.02.2020
5		Download...	
6			

Frontend 4: Microsoft Excel...

```
'Ergebnis in die Tabelle Daten eintragen:
zeilen = Split(response, Chr(10))
zeilenAnzahl = UBound(zeilen)
Sheets("Daten").Cells.Clear
For i = 0 To zeilenAnzahl
    zellen = Split(zeilen(i), ";")
    zellenAnzahl = UBound(zellen)
    If UBound(zellen) > 0 Then
        For j = 0 To zellenAnzahl
            Sheets("Daten").Cells(i + 1, j + 1).NumberFormat = "@"
            Sheets("Daten").Cells(i + 1, j + 1).Value = zellen(j)
        Next j
    End If
Next i
End Sub
```

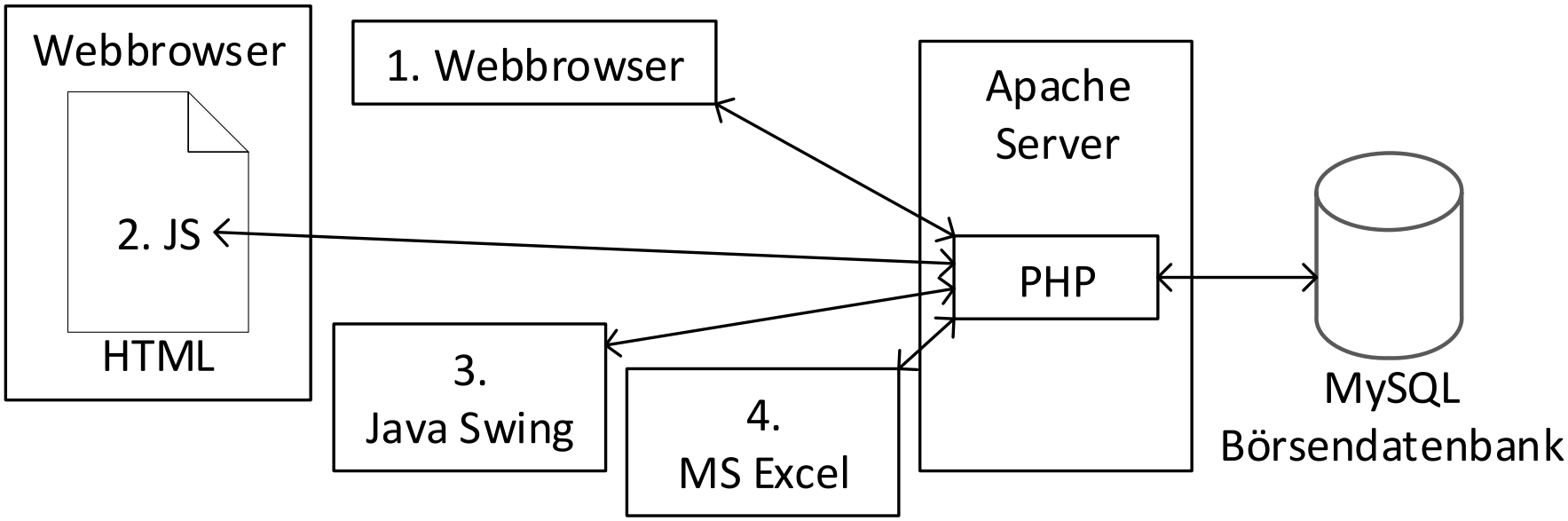
Frontend 4: Microsoft Excel...

	A	B	C	D	E	F
1	Name:	Daimler				
2	ISIN:	DE0007100000				
3	von:	02.01.2018				
4	bis:	28.02.2020				
5	Einheit:	EUR				
6	Handelsplatz:	Xetra				
7	ist Index:	nein				
8	ist ETP:	nein				
9	ist Devise:	nein				
10	datum	open	close	high	low	volumen
11	02.01.2018	70,74	70,62	71,08	69,33	279901184
12	03.01.2018	70,75	71,19	71,55	70,68	207582320
13	04.01.2018	71,8	72,02	72,5	71,74	262169152
14	05.01.2018	72,25	72,98	72,98	72,16	254139248

Eingabe | Daten

Fazit zum REST-Code

- Jede Kommunikation geht von den Clients aus:
  Der PHP-Server ist passiv & sendet nur nach Aufforderung (Request).
- Das ist gut für die Abfrage von schwach veränderlichen Daten:
  Vermeiden Sie Polling durch ständigen, automatisierten Reload!
- Die Kopplung zwischen Frontend und Backend ist sehr schwach:
  Das ist gut! 4 Frontend-Beispiele!



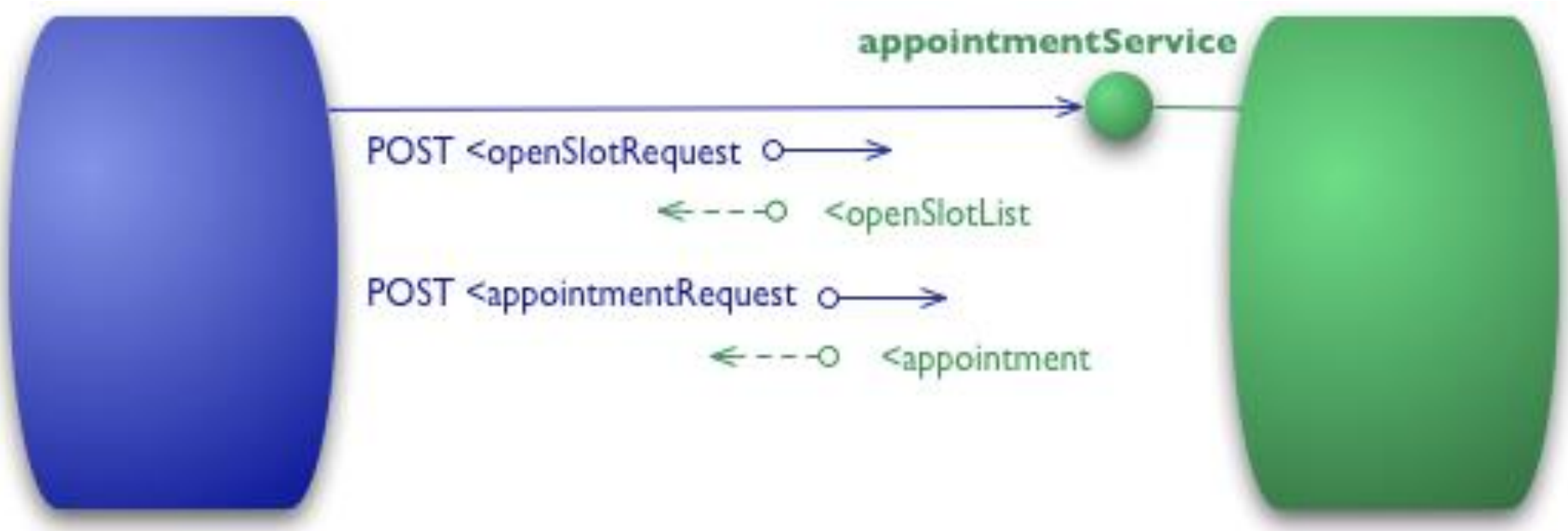
REST-Güteklassen

Das Richardson-Reifegradmodell (RMM) ist ein von Leonard Richardson entwickelter Maßstab, der angibt, wie strikt ein Service REST implementiert:

Level	Eigenschaften
0	• verwendet XML-RPC oder SOAP • der Service wird über einen einzelnen URI adressiert • verwendet eine einzelne HTTP-Methode (oft POST)
1	• verwendet verschiedene URIs und Ressourcen • verwendet eine einzelne HTTP-Methode (oft POST)
2	• verwendet verschiedene URIs und Ressourcen • verwendet mehrere HTTP-Methoden
3	• basiert auf HATEOAS und verwendet daher Hypermedia für Navigation • verwendet verschiedene URIs und Ressourcen • verwendet mehrere HTTP-Methoden

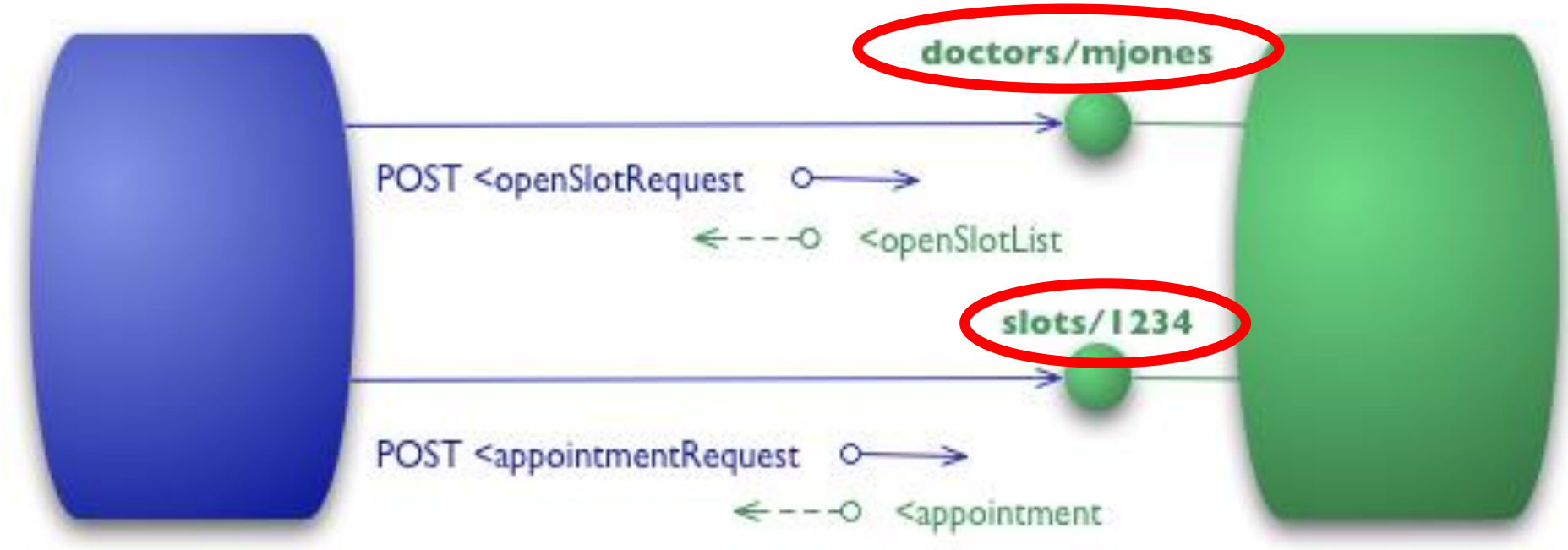
https://martinfowler.com/articles/richardsonMaturityModel.html

REST Level 0



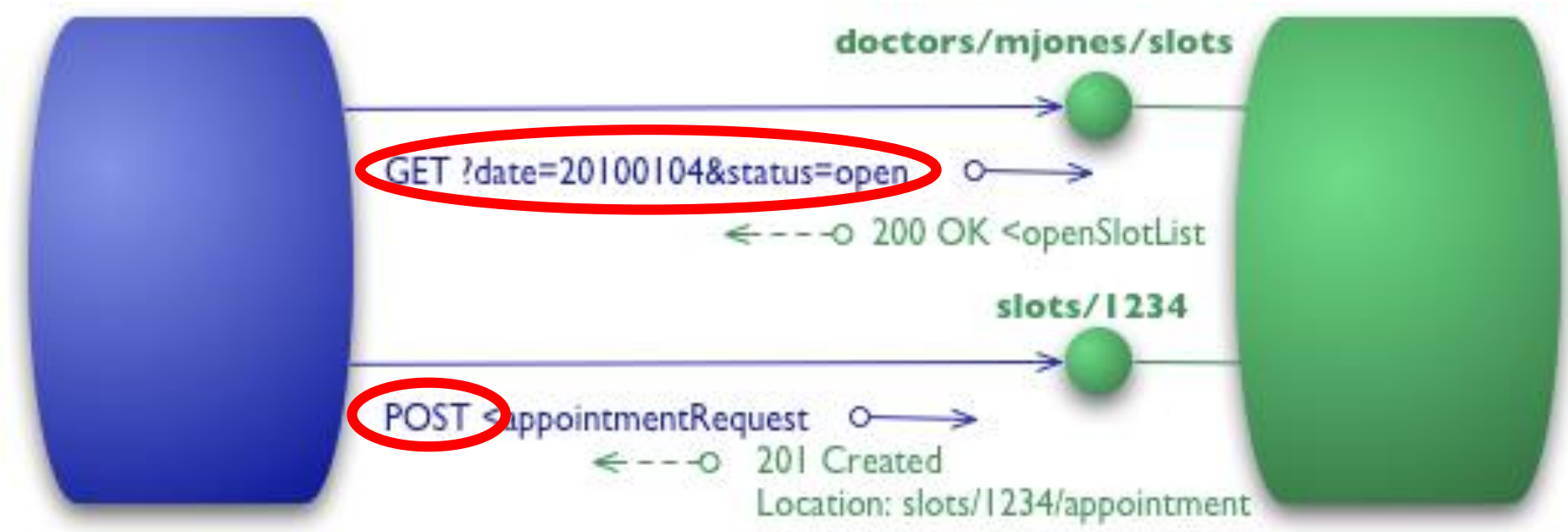
https://martinfowler.com/articles/richardsonMaturityModel.html

REST Level 1: Verschiedene URIs & Resourcen



https://martinfowler.com/articles/richardsonMaturityModel.html

REST Level 2: mehrere HTTP-Methoden



https://martinfowler.com/articles/richardsonMaturityModel.html

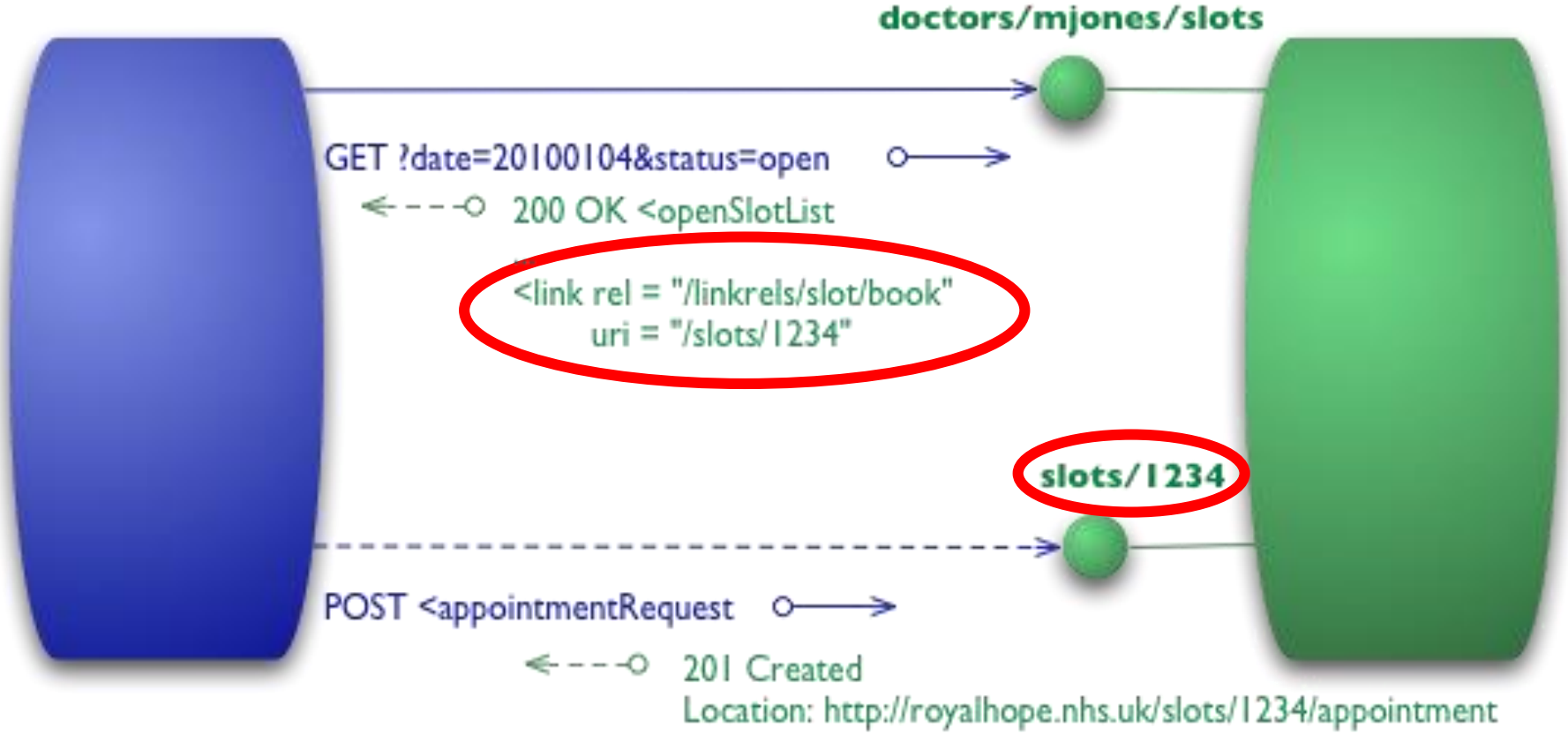
REST Level 3: Was ist HATEOAS?

- Hypermedia as the Engine of Application State (HATEOAS) ist ein Entwurfsprinzip von REST-Architekturen.
- Bei HATEOAS navigiert der Client einer REST-Schnittstelle ausschließlich über URLs, welche vom Server bereitgestellt werden.
- Abhängig von der gewählten Repräsentation geschieht die Bereitstellung der URIs über Hypermedia, also z. B.
  - in Form von „href“- und „src“-Attributen bei HTML-Dokumenten bzw. HTML-Snippets, oder
  - in für die jeweilige Schnittstelle definierten und dokumentierten JSON- bzw. XML-Attributen/-Elementen.

REST Level 3: Was ist HATEOAS?

- Abstrakt betrachtet stellen HATEOAS-konforme REST-Services einen endlichen Automaten dar, dessen Zustandsveränderungen durch die Navigation mittels der bereitgestellten URIs erfolgt.
- Durch HATEOAS ist eine lose Bindung gewährleistet und die Schnittstelle kann verändert werden.

REST Level 3: Verwendung von HATEOAS



https://martinfowler.com/articles/richardsonMaturityModel.html