

REST: I don't Think it Means What You Think it Does

Stefan Tilkov | @stilkov

GOTO Amsterdam, 20 June 2014

REST: An architectural style defined by the constraints Client-Server, Stateless Communication, Caching, Uniform Interface, Layered System, and (optionally) Code-on-demand.

For details, see <http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>

Thank you!

REST Intro

*identification
of resources*

*identification
of resources*

Universal identity

URIs

Link targets

Bookmarks

Entry points

*resource manipulation
through representations*

*resource manipulation
through representations*

SoC data/identity/location

**Multiple
representations**

Metadata

Reduced dependency

Processing model

*hypermedia as the engine
of application state*

*hypermedia as the engine
of application state*

Links

Connections

Forms

Flow

Status

self-descriptive messages

self-descriptive messages

Standardization

Metadata

Provider independence

Intermediaries

Caching

What's in the Web?

Lots of things with URLs ...

... connected with hypermedia ...

... governed by standard formats.

Common Misconceptions

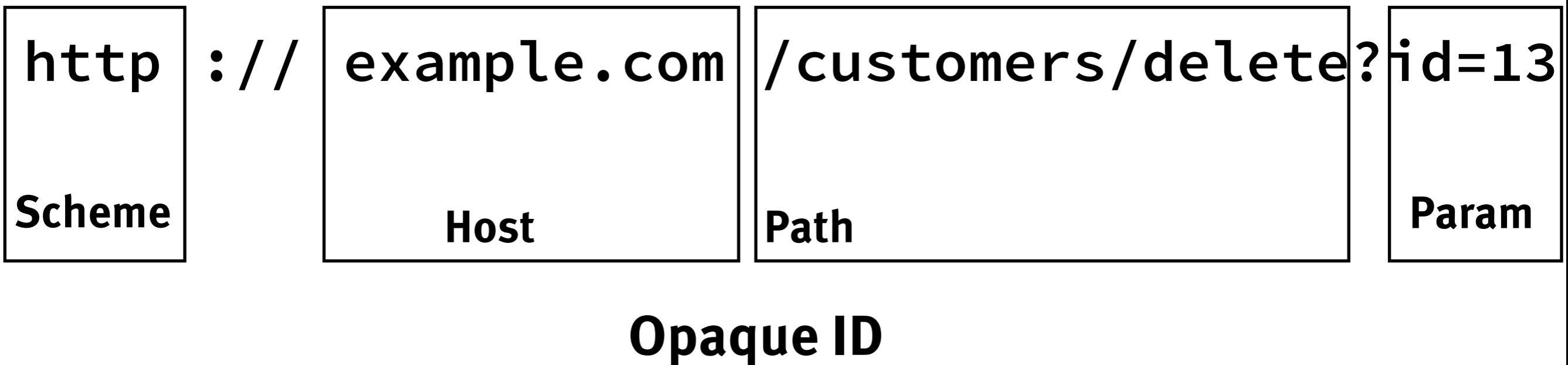
REST is about nice URIs

IT'S NOT

What makes a URI “RESTful”?

1. `http://example.com/customers/format?drive=c`
2. `http://example.com/customers/getDetails?id=13`
3. `http://example.com/customers/delete?id=13`
4. `http://example.com/customers/13`
5. `http://example.com/customers/13/edit`

There is no such thing as a “RESTful URI”



Why you shouldn't care about URIs

```
.....  
: <customer>  
:
```

```
:   <...>  
:
```

```
:   <orders href='http://example.com/orders/'>  
:
```

```
: </customer>  
:.....
```

.....
· Hypermedia context

**REST = URI patterns +
GET, PUT,
POST, DELETE**

CLOSE, BUT NO

URI	Documented URIs become APIs	Method	Meaning
http://ex.org/v1/customers			POST
http://ex.org/v1/customers/{id}		GET	get customer details
http://ex.org/v1/customers/{id}/orders			get list of customer's details
...	Versions in URIs cause change for no good reason	Assumptions about server details become facts	

Detour: Versioning



What are you “versioning”?

Data ✓

Documentation ✓

Formats ✓

APIs ✗

Versioning recommendations

*Use different media types
for different things*

*Use version numbers in URIs to
version the resource itself*

*Create new resources
for new aspects*

Reserve space for links

Version your docs

Wait, what?

Yes, no versioning

Postel's law

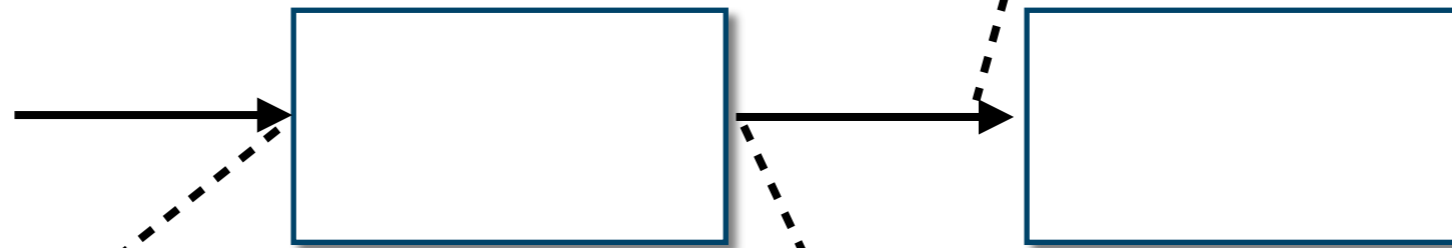
“TCP implementations should follow a general principle of robustness: Be conservative in what you do, be liberal in what you accept from others.”

<http://tools.ietf.org/html/rfc761>

Be kind to each other

Postel's Law

Liberal, Loose, Tolerant



Liberal, Loose, Tolerant

Strict, Critical, Draconian

Client rules

**Don't depend on
URI structure**

**Support unknown
links**

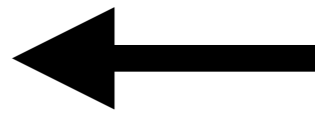
Ignore unknown content

Server rules

**Don't break URI structure
unnecessarily**

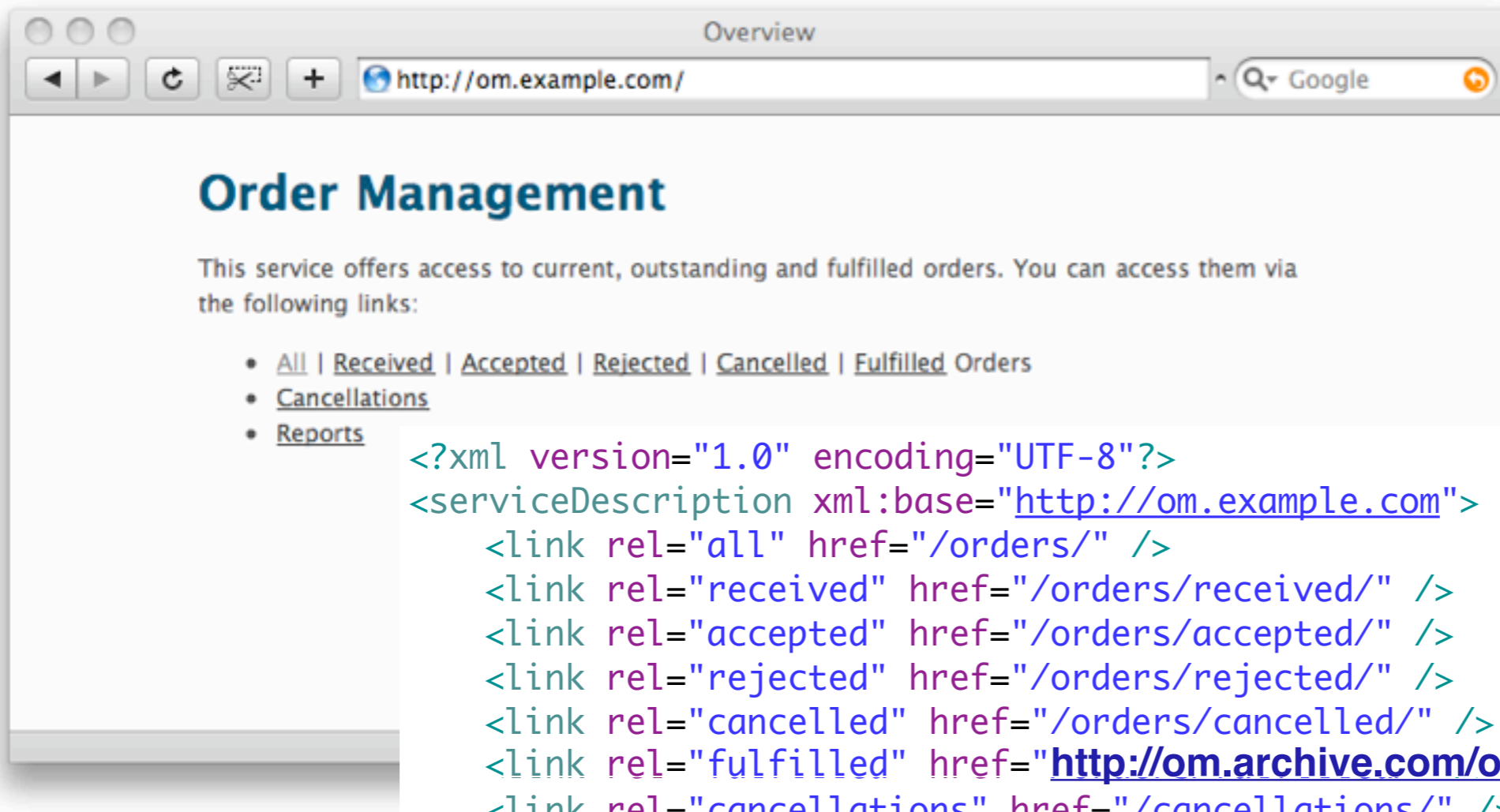
**Evolve via additional
resources**

Support older formats



How to hypermedia-enable your service interfaces

Step 1: Service Documents



```
<?xml version="1.0" encoding="UTF-8"?>
<serviceDescription xml:base="http://om.example.com">
  <link rel="all" href="/orders/" />
  <link rel="received" href="/orders/received/" />
  <link rel="accepted" href="/orders/accepted/" />
  <link rel="rejected" href="/orders/rejected/" />
  <link rel="cancelled" href="/orders/cancelled/" />
  <link rel="fulfilled" href="http://om.archive.com/orders/" />
  <link rel="cancellations" href="/cancellations/" />
  <link rel="reports" href="/reports/" />
</serviceDescription>
```

```
{ "serviceDescription" : {
  "base": "http://om.example.com",
  "links": [
    { "rel": "all", "href": "/orders/" },
    { "rel": "all", "href": "/orders/" },
    { "rel": "received", "href": "/orders/received/" },
    { "rel": "accepted", "href": "/orders/accepted/" },
    { "rel": "rejected", "href": "/orders/rejected/" },
    { "rel": "cancelled", "href": "/orders/cancelled/" },
    { "rel": "fulfilled", "href": "/orders/fulfilled/" },
    { "rel": "cancellations", "href": "/cancellations/" },
    { "rel": "reports", "href": "/reports/" }
  ]
}
```

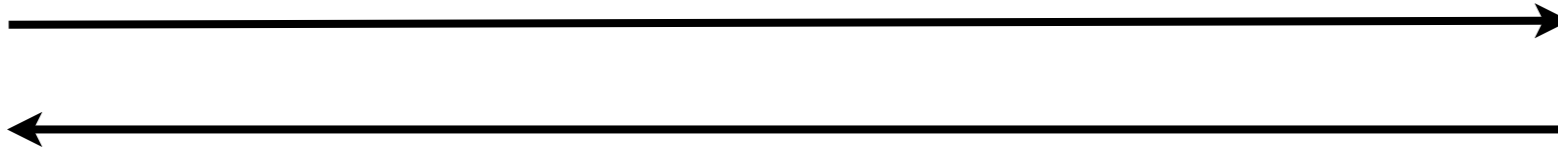

JSON Home

```
{
  "resources": {
    "http://example.org/rel/widgets": {
      "href": "/widgets/"
    },
    "http://example.org/rel/widget": {
      "href-template": "/widgets/{widget_id}",
      "href-vars": {
        "widget_id": "http://example.org/param/widget"
      },
      "hints": {
        "allow": ["GET", "PUT", "DELETE", "PATCH"],
        "representations": ["application/json"],
        "accept-patch": ["application/json-patch"],
        "accept-post": ["application/xml"],
        "accept-ranges": ["bytes"]
      }
    }
  }
}
```

<http://tools.ietf.org/html/draft-nottingham-json-home-03>

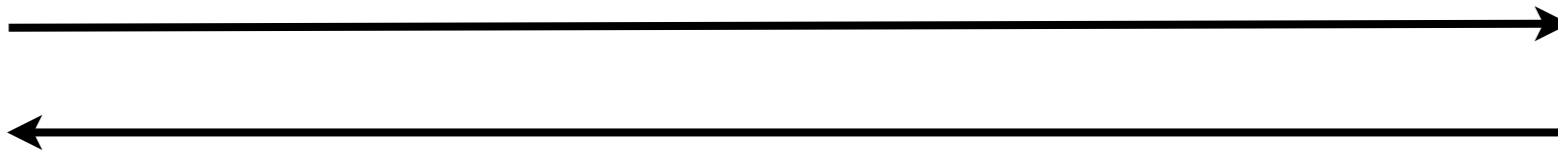
Step 2: Resource Links

GET /order/123



```
{
  "order": {
    "date": "2013-07-02",
    "total": 1240.02,
    "...": "...",
    "links": [
      {"rel": "self", "href": "http://example.com/orders/123"},
      {"rel": "contacts", "href": "http://example.org/A3BEF1"},
      ...
    ]
  }
}
```

GET /order/123

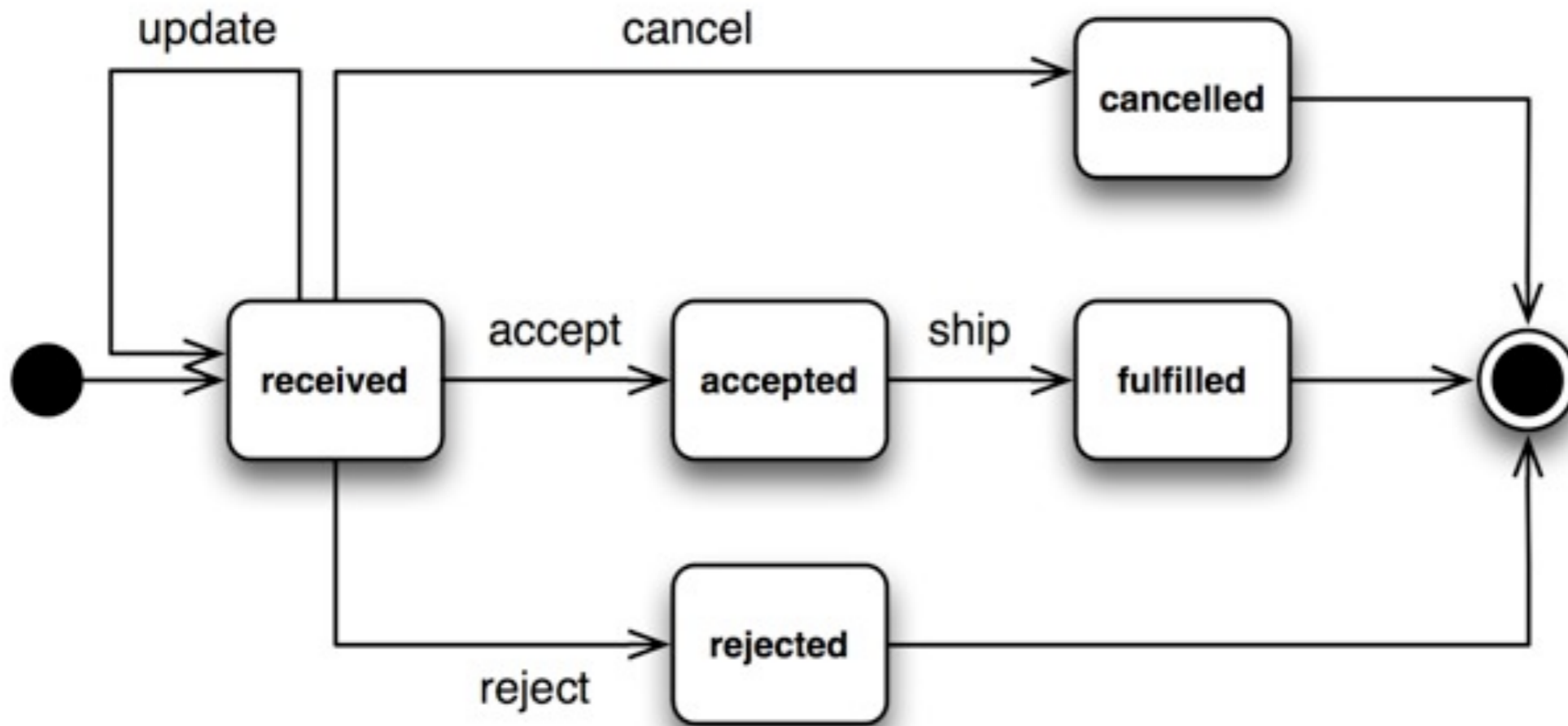


```
{  
  "order": {  
    "date": "2013-07-02",  
    "total": 1240.02,  
    "...": "...",  
    "links": {  
      "self": "http://example.com/orders/123",  
      "contacts": "http://example.org/A3BEF1",  
      ...  
    }  
  }  
}
```



Your future extensions

Step 3: State Transition Links



```

{
  "order": {
    "state" : "received",
    "...": "...",
    "links" : [
      {"rel" : "cancel", "href" : "http://..."},
      {"rel" : "accept", "href" : "http://..."},
      {"rel" : "reject", "href" : "http://..."},
      ...
    ]
  }
}

```

Hypermedia APIs = Responses with Links

NOT ONLY

*Rule #1: Don't have clients build URIs
using string concatenation*

...instead: provide recipes

```
<form action='http://example.com/search' method='GET'>  
  Search for: <input type='text' name='query'>  
              <input type='submit'>  
</form>
```

```
{  
  "rel": "search",  
  "template": "http://example.com/search?q={query}"  
}
```

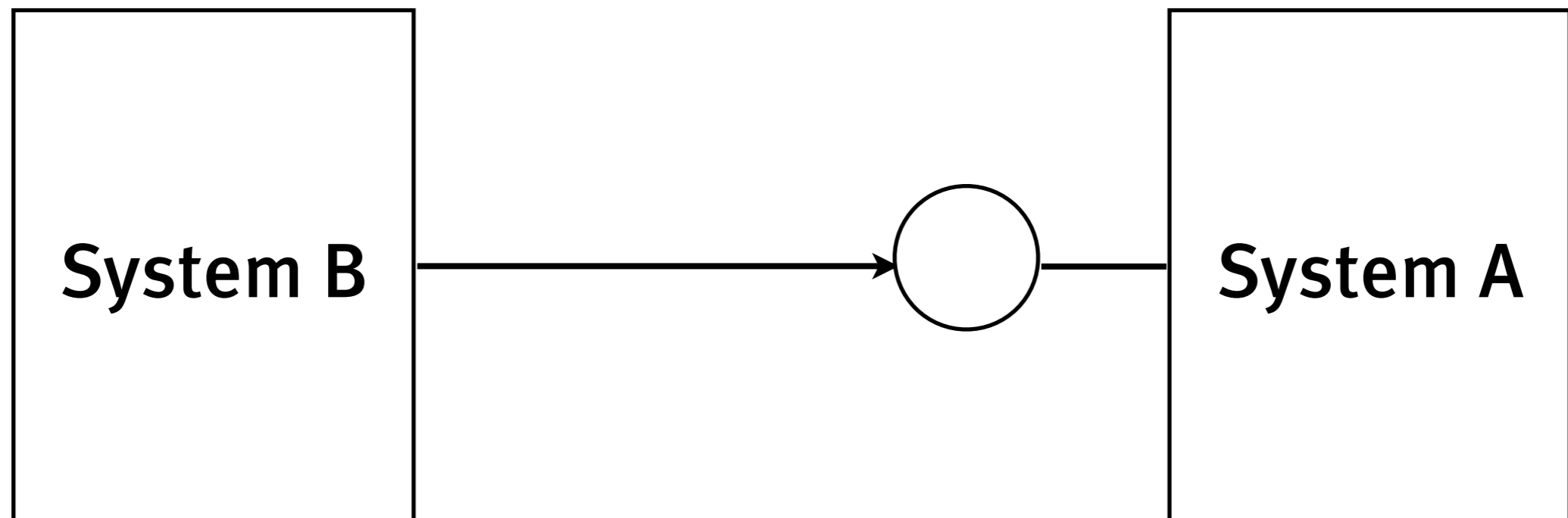
```
<form action='http://example.com/add' method='POST'>
  First:    <input type='text' name='first'>
  Last:     <input type='text' name='last'>
  Birthday: <input type='date' name='bdate'>
            <input type='submit'>
</form>
```

```
{"target": "http://example.com/add",
 "rel": "add",
 "template": {
   "first": "...",
   "last": "...",
   "bdate": "...
 }
}
```

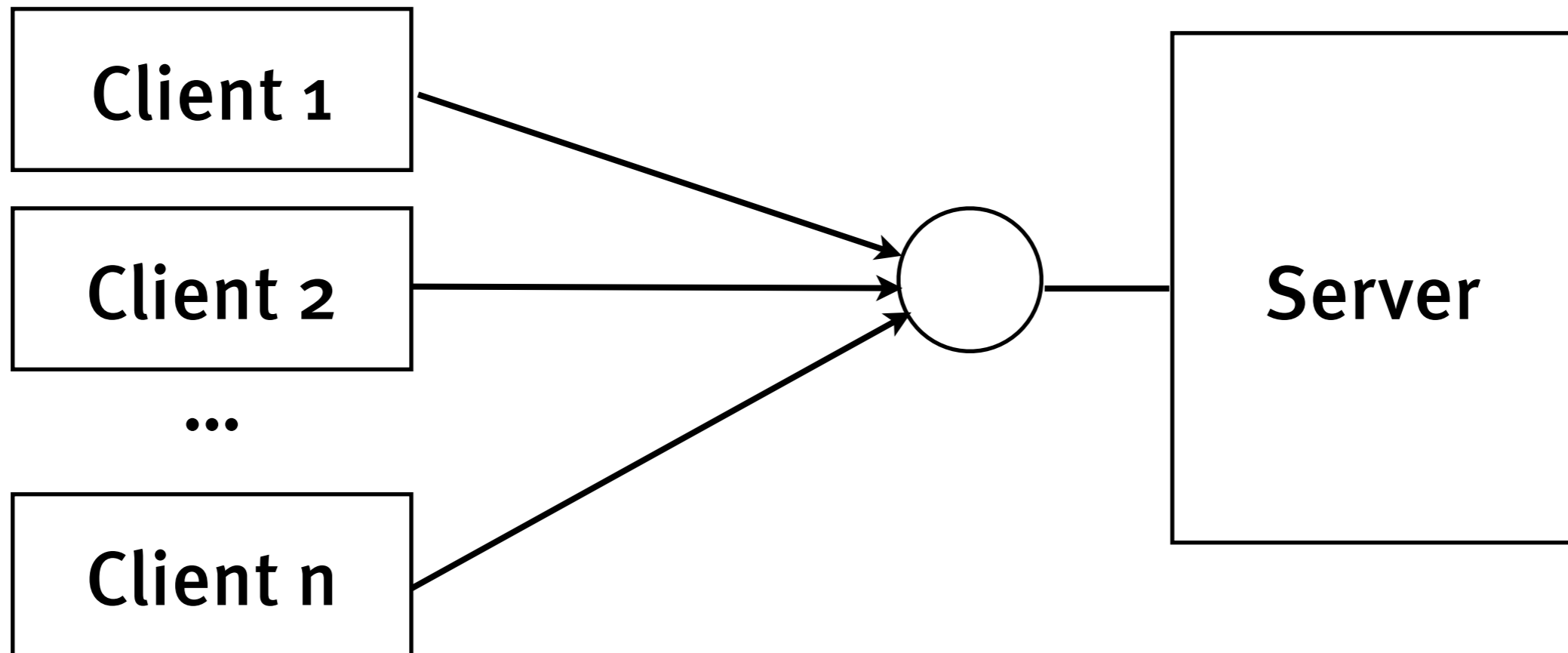
**REST = A different approach to
service interfaces**

TOO SIMPLE

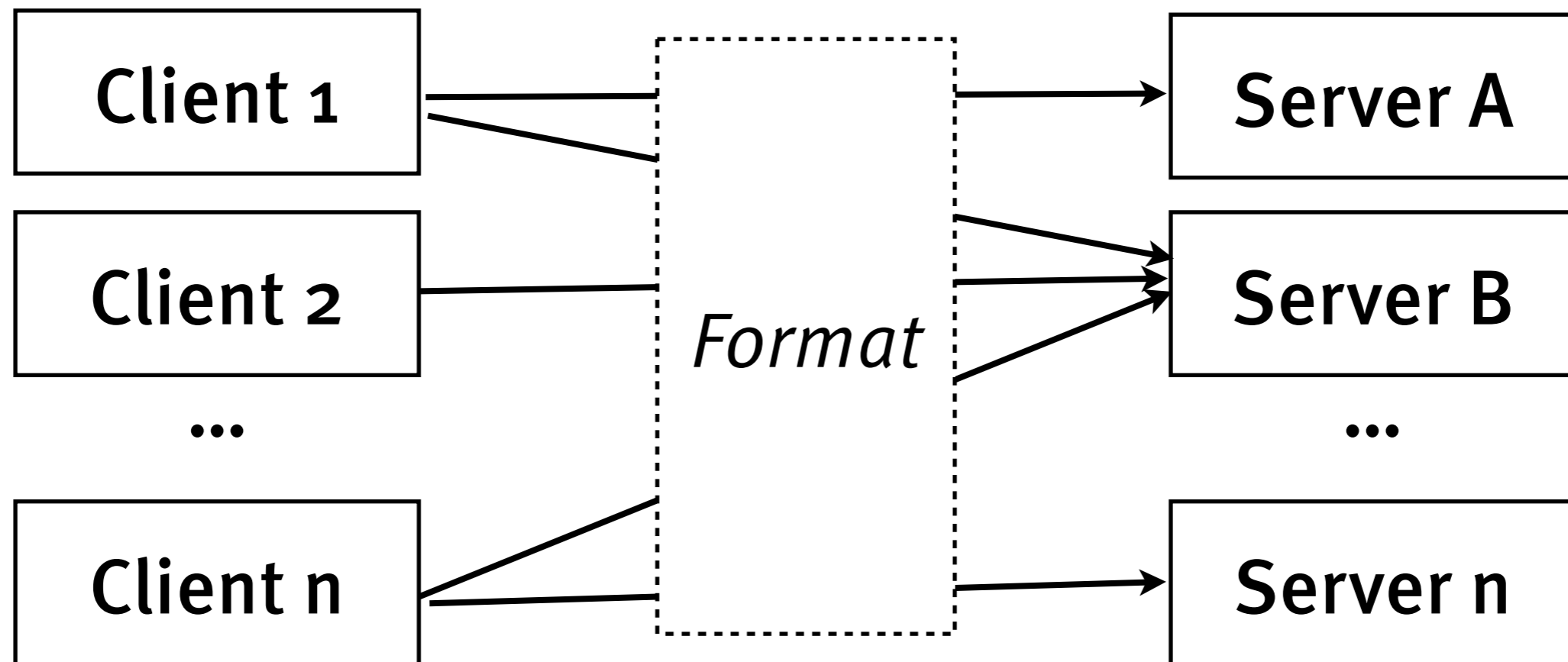
Point-to-point integration



Service interfaces



Hypermedia types



application/atom+xml

application/opensearchdescription+xml

application/vnd.collection+json

application/hal+json

application/vnd.siren+json

text/html

Detour: HTML



```
<order>
  <shippingAddress>Paris, France</shippingAddress>
  <items>
    <item>
      <productDescription>iPad</productDescription>
      <quantity>1</quantity>
      <price>699</price>
    </item>
  </items>
  <link href="http://om.example.com/cancellations"
        rel="cancel" />
  <link href="https://om.example.com/orders/123/payment"
        rel="payment" />
</order>
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<body>
  <div class="order">
    <p class="shippingAddress">Paris, France</p>
    <ul class="items">
      <li class="item">
        <p class="productDescription">iPad</p>
        <p class="quantity">1</p>
        <p class="price">699</p>
      </li>
    </ul>
    <a href="http://om.example.com/cancellations"
      rel="cancel">cancel</a>
    <a href="https://om.example.com/orders/123/payment"
      rel="payment">payment</a>
  </div>
</body>
</html>
```


Benefits of Using HTML as Your Hypermedia Format

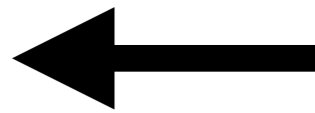
Ubiquity

Well-known, well supported HM controls

A pretty good standard client

Lots of programming tools

UIs as a side-effect



Summary

Link context over pretty URIs

Hypermedia over URI APIs

Hypermedia flows over static links

Generalized formats over services



www.innoq.com

innoQ Deutschland GmbH

Krischerstr. 100
40789 Monheim am Rhein
Germany
Phone: +49 2173 3366-0

Ohlauer Straße 43
10999 Berlin
Germany
Phone: +49 2173 3366-0

Robert-Bosch-Straße 7
64293 Darmstadt
Germany
Phone: +49 2173 3366-0

Radlkoferstraße 2
D-81373 München
Telefon +49 (0) 89 741185-270

innoQ Schweiz GmbH

Gewerbestr. 11
CH-6330 Cham
Switzerland
Phone: +41 41 743 0116

info@innoq.com

Q&A

Stefan Tilkov, @stilkov
stefan.tilkov@innoq.com
Phone: +49 170 471 2625