



What needs to be done to make the REST architectural style clear on the notion that hypertext is a constraint? In other words, if the engine of application state (and hence the API) is not being driven by hypertext, then it cannot be RESTful and cannot be a REST API. Period. Is there some broken manual somewhere that needs to be fixed?

– Roy T. Fielding

REST Assured

Hypermedia APIs with Spring

Oliver Gierke



Oliver Gierke

Engineer @ Pivotal

Spring Data project lead



✉ ogierke@gopivotal.com

🌐 www.olivergierke.de

🐦 [olivergierke](https://twitter.com/olivergierke)

Background

REST

REST

Resources

URIs

Uniform Interface

Representations

Hypermedia

Hypermedia



*HATEOAS - the word,
there's no pronunciation for.*

- Ben Hale (SpringOne2GX 2012)

Hypermedia

Links in representations

State navigations discoverable

REST in practice

/THEORY/IN/PRACTICE

REST in Practice

Hypermedia and Systems Architecture

Jim Webber
Savas Parastatidis
Ian Robinson

O'REILLY®

O'REILLY®

Ian Robinson
Savas Parastatidis
Jim Webber

RESTBucks

RESTBucks

Starbucks (like) coffee ordering

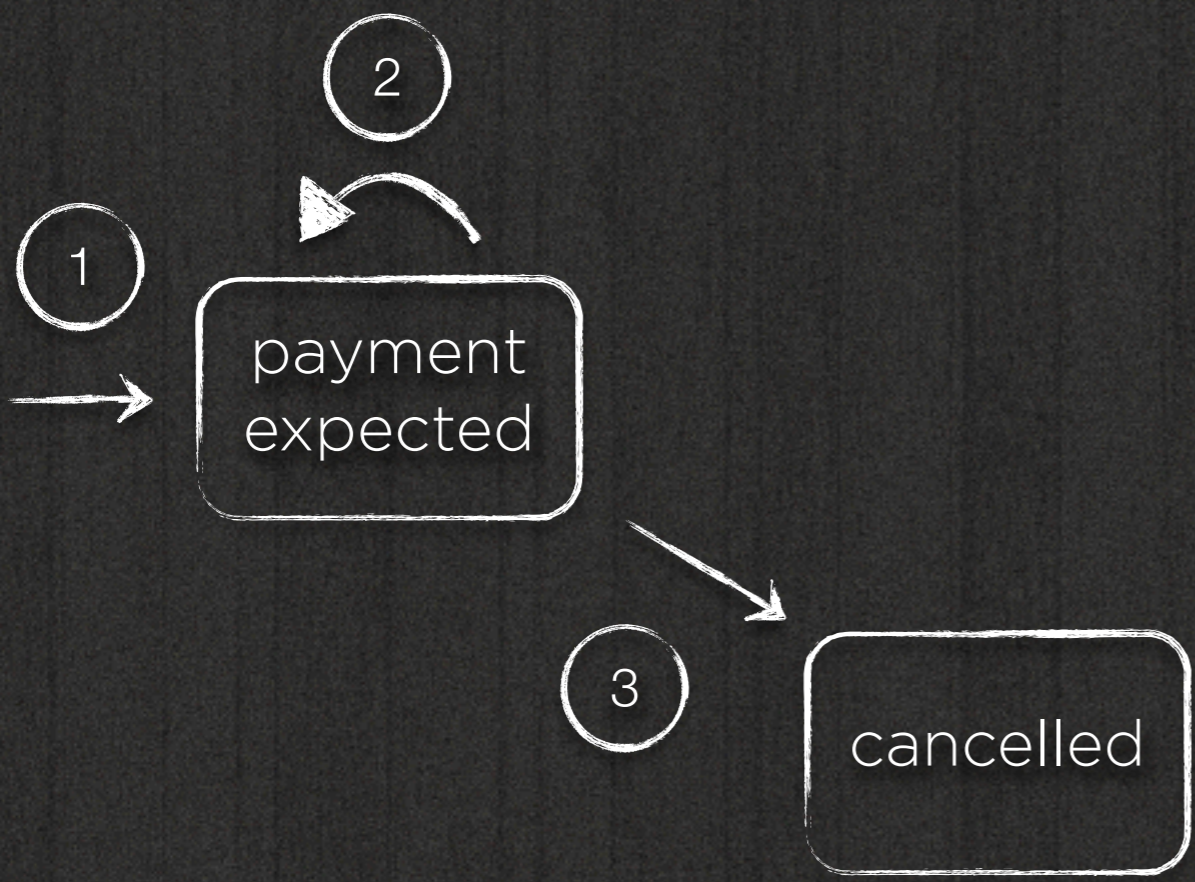
Order / Payment

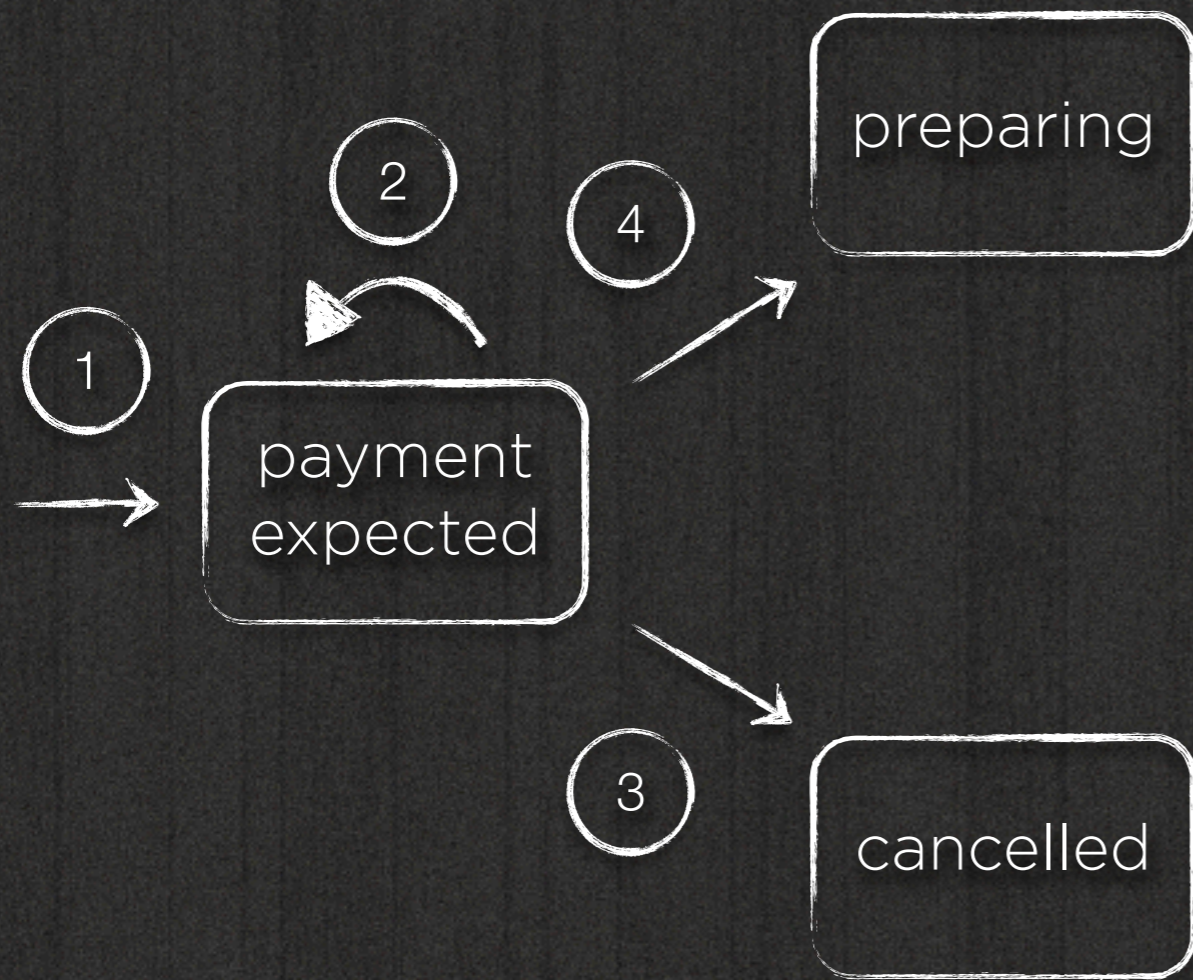
1

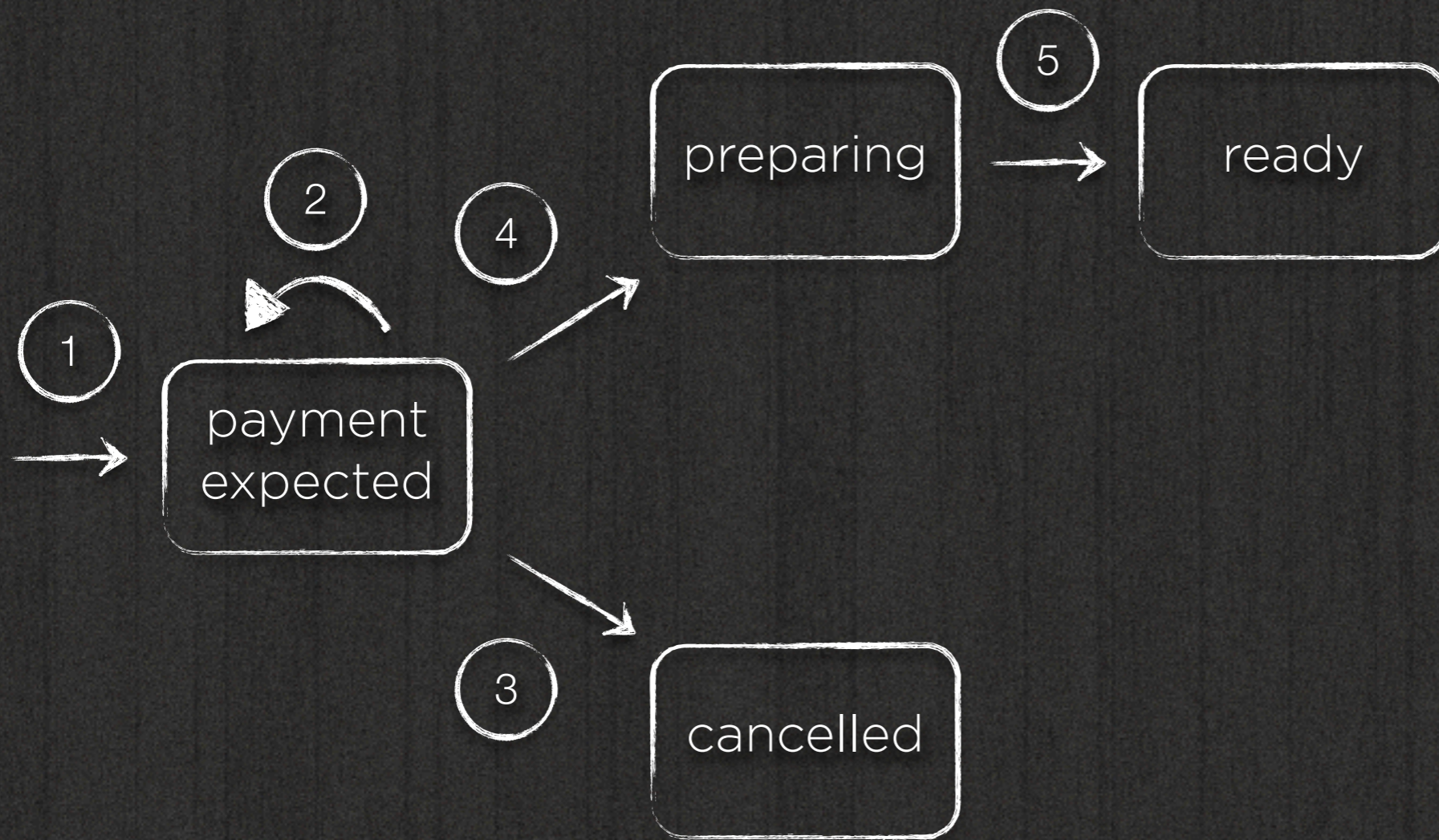


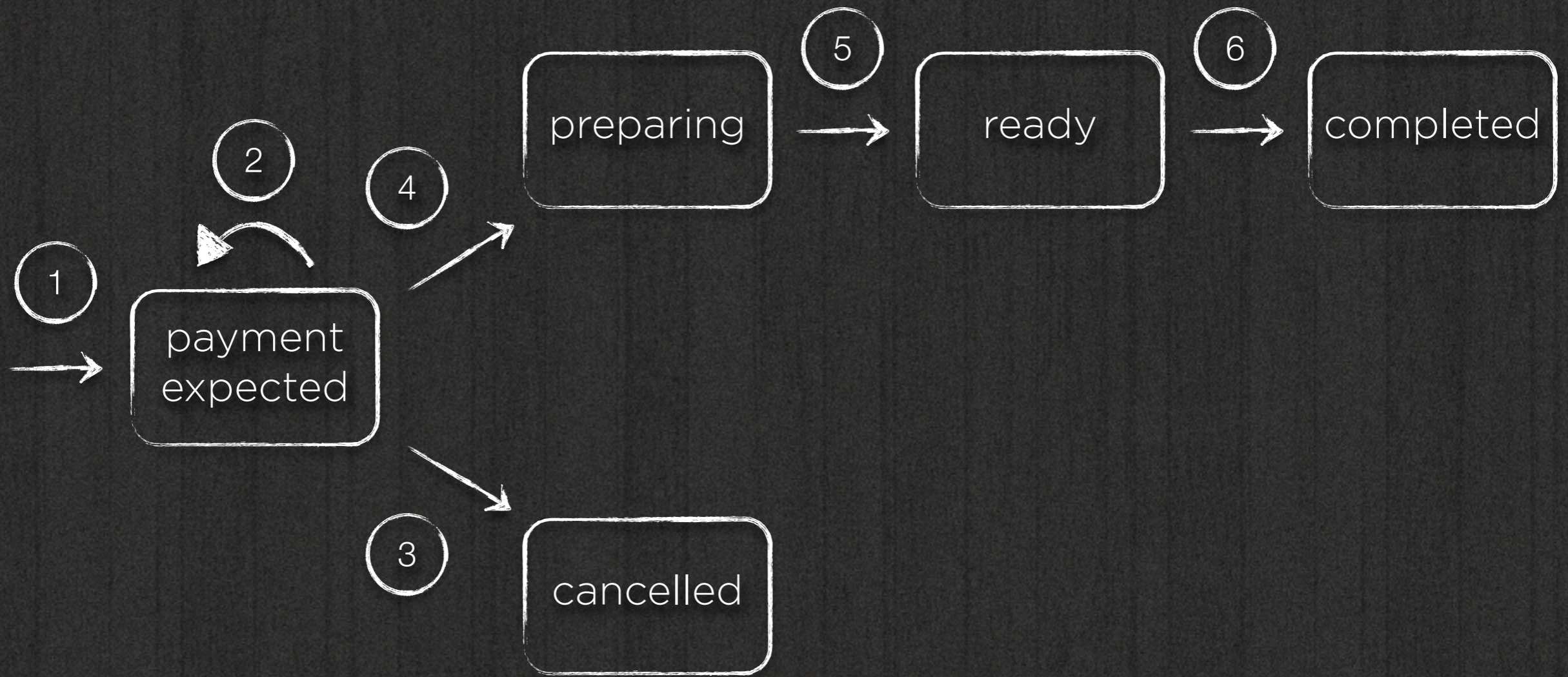
payment
expected











Method	URI	Action	Step
POST	/orders	Create new order	1
POST/PATCH	/orders/4711	Update the order (only if "payment expected")	2
DELETE	/orders/4711	Cancel order (only if "payment expected")	3
PUT	/orders/4711/payment	Pay order (only if "payment expected")	4

Barista preparing the order

GET	/orders/4711	Poll order state	5
GET	/orders/4711/receipt	Access receipt	
DELETE	/orders/4711/receipt	Conclude the order process	6

Challenges

Challenges

How to avoid hard coding URIs?

Use link
relations

orders	Returns all orders available in the system
order	Returns a single order
self	The uri value can be used to GET the latest resource representation of the order.
cancel	This is the URI to be used to DELETE the order resource should the consumer wish to cancel the order.
update	Consumers can change the order using a POST to transfer a representation to the linked resource.
payment	The linked resource allows the consumer to begin paying for an order. Initiating payment involves PUTting an appropriate resource representation to the specified URI.
receipt	The URI to access the receipt using GET and conclude the order by taking the receipt (use DELETE).

orders	Returns all orders available in the system
order	Returns a single order
self	The uri value can be used to GET the latest resource representation of the order.
cancel	This is the URI to be used to DELETE the order resource should the consumer wish to cancel the order.
update	Consumers can change the order using a POST to transfer a representation to the linked resource.
payment	The linked resource allows the consumer to begin paying for an order. Initiating payment involves PUTting an appropriate resource representation to the specified URI.
receipt	The URI to access the receipt using GET and conclude the order by taking the receipt (use DELETE).

Method	URI	Action	Step
POST	/orders	Create new order	1
POST/PATCH	/orders/4711	Update the order (only if "payment expected")	2
DELETE	/orders/4711	Cancel order (only if "payment expected")	3
PUT	/orders/4711/payment	Pay order (only if "payment expected")	4

Barista preparing the order

GET	/orders/4711	Poll order state	5
GET	/orders/4711/receipt	Access receipt	
DELETE	/orders/4711/receipt	Conclude the order process	6

Method	Relation type	Action	Step
POST	orders	Create new order	1
POST/PATCH	update	Update the order (only if "payment expected")	2
DELETE	cancel	Cancel order (only if "payment expected")	3
PUT	payment	Pay order (only if "payment expected")	4

Barista preparing the order

GET	order	Poll order state	5
GET	receipt	Access receipt	
DELETE	receipt	Conclude the order process	6

Challenges

How to implement:
"only if payment expected"?

Process modeling

Root resource

The only URI known

GET /

```
{ _links : [ orders : { href : ".../orders" } ] }
```


Place order

Access root resource

Follow `orders` link

`$_links.orders.href`

`POST /orders`

```
{ _links : [ orders : { href : ".../orders" } ] }
```

```
{ _links : [ { self : { href : ... } },
              { cancel : { href : ... } },
              { update : { href : ... } },
              { payment : {
                  href : ".../orders/4711/payment" }} ],
  _embedded : {
    order : {
      items : [ {
        ...
      } ],
      location : "take-away",
      price : 4.2,
      status : "payment expected"
    }
  }
}
```

Trigger payment

Follow `payment` link

```
$_links.payment.href
```

```
PUT /orders/4711/payment
```

```
{ _links : [
  { self : { href : ".../orders/4711" } }, ...
  { payment : { href : ".../orders/4711/payment" } } ],
  _embedded : {
    order : {
      items : [ {
        ...
      } ],
      location : "take-away",
      price : 4.2,
      status : "payment expected"
    }
  }
}
```

```
{ _links : [  
  { self : { href : ".../orders/4711/payment" } },  
  { order : { href : ".../orders/4711" } } ],  
  
  _embedded : {  
    payment : {  
      creditCard : { ... },  
      amount : {  
        currency : "EUR",  
        value : 4.2  
      }  
    }  
  }  
}
```

Poll order

Follow `order` link

```
$_links.order.href
```

```
GET /orders/4711
```

```
ETag / If-None-Match
```

```
{ _links : [  
  { self : { href : ".../orders/4711/payment" } },  
  { order : { href : ".../orders/4711" } } ],  
  
  _embedded : {  
    payment : {  
      creditCard : { ... },  
      amount : {  
        currency : "EUR",  
        value : 4.2  
      }  
    }  
  }  
}
```



```
{ _links : [
  { self : { href : ".../orders/4711" } } ],
  _embedded : {
    order : {
      items : [ {
        ...
      } ],
      location : "take-away",
      price : 4.2,
      status : "preparing"
    }
  }
}
```

Poll order

Use caching

ETag / If-None-Match

until...

```
{ _links : [
  { self : { href : ".../orders/4711" } }, ...
  { receipt : { href : ".../orders/4711/receipt" } } ],

  _embedded : {
    order : {
      items : [ {
        ...
      } ],
      location : "take-away",
      price : 4.2,
      status : "payment expected"
    }
  }
}
```

Access receipt

Follow `receipt` link

`$_links.receipt.href`

`GET /orders/4711/receipt`

```
{ _links : [
  { self : { href : ".../orders/4711" } }, ...
  { receipt : { href : ".../orders/4711/receipt" } } ],
  _embedded : {
    order : {
      items : [ {
        ...
      } ],
      location : "take-away",
      price : 4.2,
      status : "payment expected"
    }
  }
}
```

Conclude order

Follow `receipt` link

`$_links.receipt.href`

`DELETE /orders/4711/receipt`

Implementation aspects

Hypermedia VS. Java Frameworks

	Spring MVC	JAX-RS
HTTP Methods	✓	✓
URI Mapping	✓	✓
Content negotiation	✓	✓
Hypermedia	?	?

Spring HATEOAS

Spring HATEOAS

Representation models

LinkBuilder API

Representation enrichment

<http://bit.ly/spring-hateoas>

DEMO

Spring RESTBucks

Spring RESTBucks

Sample implementation

Using Spring technologies

<http://bit.ly/spring-restbucks>

	Orders	Payment
Web	Spring Data REST	Manual implementation
Service	-	Manual implementation
Repository	Spring Data	Spring Data

DEMO

API docs?

Services for a Changing World

RESTful Web APIs



O'REILLY®

*Leonard Richardson,
Mike Amundsen & Sam Ruby*

O'REILLY®

*Mike Amundsen & Sam Ruby
Leonard Richardson*



How does the client make sense of all this?

Profiles

Profile

RFC 6906

Points to resources describing additional semantics within a media type

Curies

(HAL) Curies

```
{ "_links" : [  
  
  { "self" : { "href" : ... } },  
  { "restbucks:cancel" : { "href" : ... } },  
  { "restbucks:update" : { "href" : ... } },  
  
  { "curies" :  
    { "name" : "restbucks",  
      "href" : ".../rels/{rel}"  
      "templated" : true  
    }  
  }  
],  
  ...  
}
```

Docs for `update` are at: `.../rels/update`

alps.io

alps.io

Application Level Profile Semantics

Describe state transitions and payloads

Media type agnostic

Repository with pre-defined docs

`application/alps+(json|xml)`

DEMO

Miscellaneous

Spring MVC integration testing

REST Shell

Thank you!

Resources

Code

Spring HATEOAS Sample

Spring RESTBucks

Spring Data REST

Spring HATEOAS

REST Shell

Books

REST in Practice

REST und HTTP

RESTful WebServices Cookbook

RESTful Web APIs

Spring Data

Videos

Hypermedia APIs - Jon Moore

Hypermedia APIs with Spring